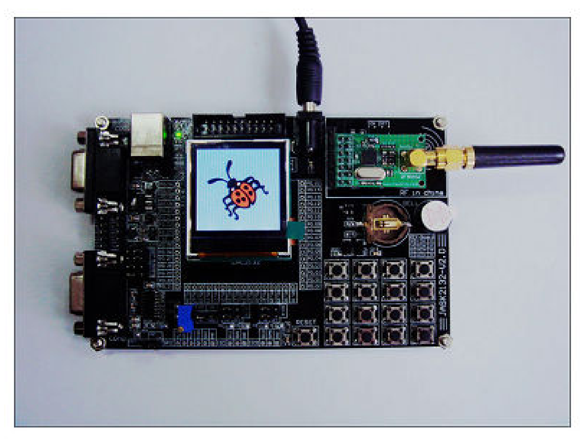
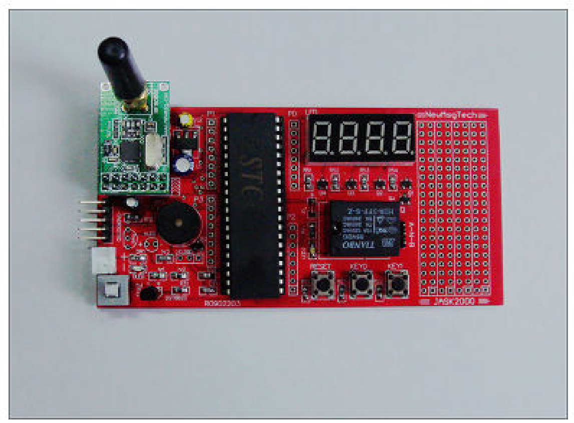
JASK2132 开发使用手册

ARM & RF 系列开发套件

V2.0

追赶时代技术潮流　　加速科技创新步伐





显著特点

1. 从 8 位单片机到 32 位 ARM7 处理器一站式轻松入门
2. 精通无线通信与嵌入式两大热门技术，做时代的强者
3. 视频教学指导，要点解析，举一反三，开发技巧高效速成
4. 基础性、启发性案例丰富，从简单到复杂，从入门到精通

公司名称：杭州威步科技有限公司　　联系电话：0571-86576692 18958007935
传真：0571-86576692　　E-mail: wenming_hu2002@yahoo.com

第一部分 嵌入式无线通信技术简介.................................... 4
功能框架示意图.. 5
如何正确认识和定位嵌入式？ 5
如何成为一名全面的嵌入式工程师？ 6
JASK2132 开发板的设计定位是什么？............................ 6
嵌入式无线通信技术的典型应用领域............................ 7
第二部分 ARM7-LPC2132 芯片解析.................................. 8
LPC213X概述.. 9
LPC213X基本特性...................................... 10
硬件功能示意图.. 12
锁相环PLL和VPB分频器要点解析.............................. 12
锁相环PLL和VPB分频器概述和意义........................ 12
四类时钟频率的区别和各自作用.......................... 13
锁相环PLL和VPB分频器基本操作方法...................... 13
PLL和VPB操作示例..................................... 14
在操作PLL和VPB时的注意事项............................ 15
中断操作要点解析.. 15
向量中断控制器（VIC）概述............................. 15
向量中断控制器（VIC）特性............................. 16
VIC基本操作方法...................................... 16
中断操作示例.. 17
VIC使用注意事项...................................... 17
GPIO操作要点解析.. 19
GPIO功能概述... 19
功能模块的操作方法.................................... 21
GPIO的读、写操作方法.................................. 21
GPIO使用注意事项...................................... 22
UART应用要点解析.. 22
UART功能概述... 22
UART基本操作方法..................................... 25
UART0 示例分析 25
I2C要点解析... 27
I2C功能概述.. 27
I2C基本特性.. 28
I2C操作模式.. 29
I2C主机基本操作方法.................................. 32
SPI模块要点解析... 33
SPI功能概述.. 33
SPI基本特性.. 33
SPI基本操作方法...................................... 33
SPI主机操作流程...................................... 36

SPI从机操作流程 36
SPI操作示例 37
定时器使用要点分析 38
定时器功能概述 38
定时器基本特性 38
定时器 0 操作示例 39
PWM脉宽调制器要点解析 39
PWM概述 39
PWM基本特性 40
PWM操作示例 45
AD采集要点解析 46
AD功能概述 46
AD基本特性 46
AD模块的寄存器功能 46
AD基本操作方法 49
AD操作示例 49
RTC时钟要点解析 51
RTC功能概述 51
RTC基本特性 51
RTC基本操作方法 52
RTC操作示例 53
RTC使用注意事项 54
第三部分 基于LPC2132 嵌入式无线应用 55
主流无线芯片汇总及特点解析 55
系列A: 433/868/915MHZ频段 55
1. NRF905 基本特性 55
2.SI4432 基本特性 56
3.CC1100 芯片特性 57
4.CC1020 芯片特性 57
5.A7102 基本特性 58
系列B: 2.4GHZ频段类 58
1.NRF24L01 芯片特性 58
2.NRF2401A基本特性 59
3.CC2500 基本特性 60
系列C: ZIGBEE协议类 60
1.CC2420/CC2520 基本特性 61
基于LPC2132 无线应用案例解析 62
案例一: 无线双向遥控 62
功能概述 62
实验准备 62
实验目的: 62

基本工作流程.. 62
操作步骤:.. 63
案例二: 无线 232/485 通信设计.................................. 64
功能概述.. 64
实验准备:.. 64
实验目的:.. 64
基本该工作流程.. 64
操作步骤:.. 65
案例三: 无线温度数据采集.................................. 65
功能概述.. 65
实验准备:.. 66
基本工作流程:.. 66
操作步骤:.. 67
案例四:无线AD传感器应用.................................. 68
功能概述.. 68
实验准备:.. 68
基本工作流程:.. 69
操作步骤:.. 69
无线应用注意事项.. 70

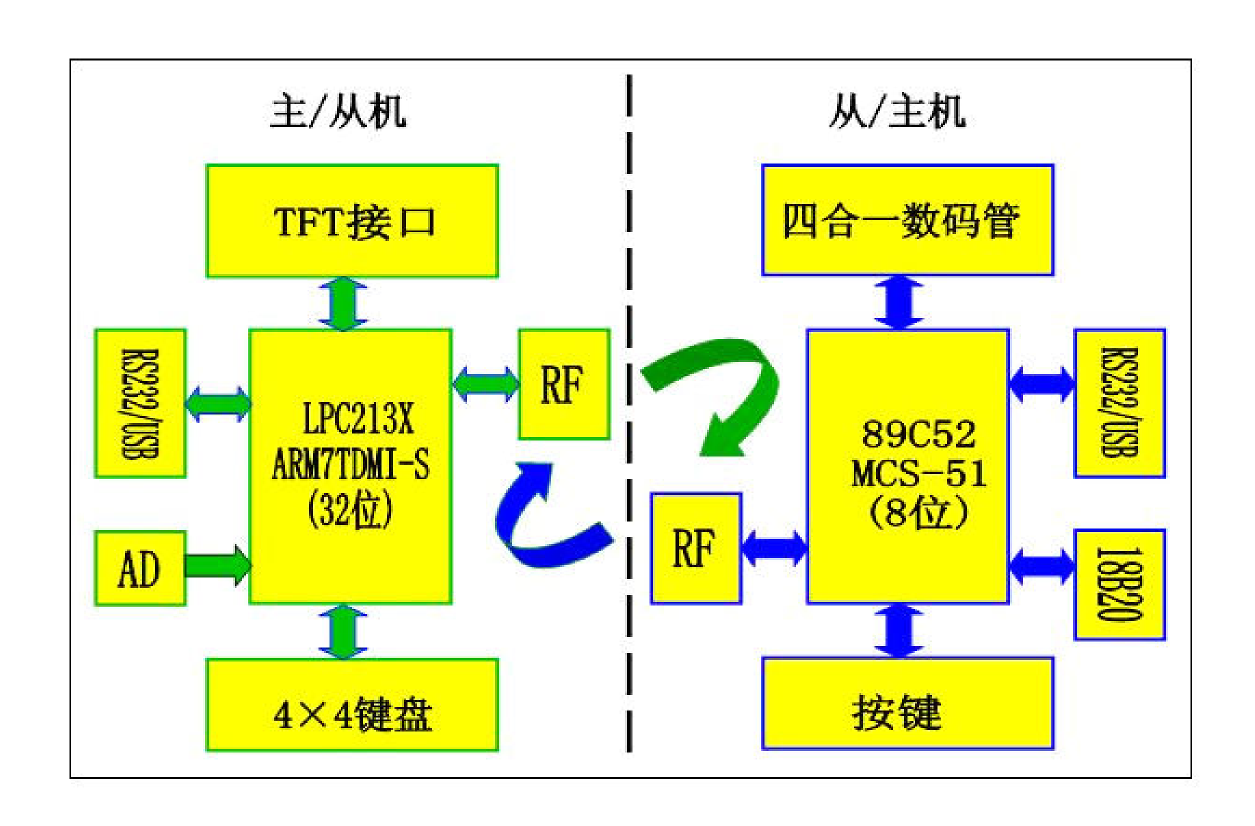
基本配置:

类型	数量	备注
JASK2132 开发底板	1 块	ARM7--LPC2132
JASK2000 开发底板	1 块	MCS51--STC89C52
TFT 准真彩显示屏	1 块	18 位像素 128X128
无线模块	2 片	可选选配如：RF905SE 等
JLINK 仿真器	1 个	USB 接口用于 ARM 调试
串口线	1 条	用于串口通信
USB-TTL 模块	2 个	用于 JASK2000 下载和串口通信
5V 电源适配器	1 个	电源输入
光盘	1 张	提供源码、入门视频等

资料丰富，设计结构系统化，及时是初学者，也一样可以 8 位机到 32 位机轻松入门，同时感受无线通信的魅力，嵌入式入门首选。

第一部分 嵌入式无线通信技术简介

功能框架示意图



如何正确认识和定位嵌入式？

经过多年的电子应用设计，从8051到AVR、PIC、430再到ARM7、ARM9，对通用MCU 的应用有了比较全面掌握和长足积累， PIC 以抗干扰能力强而广受好评，430是低功耗设计首选，ARM在处理能力上有了质的飞跃，也许很多人瞧不起8051，但

从 8051 的问世至今，以其最丰富的资料、最广泛的使用人群，入门容易，且价格也相对较低等优势，仍然是常青树。所以，不同的应用，在考虑设计成本、抗干扰性、低功耗、处理速度等综合性能的基础上，不得不承认任何一种 MCU 都无法完全取代其他。认识和分析任何一个实物，在横向特性的对比的同时，同样需要纵向的性能的权衡，唯有这样嵌入式系统开发才有效而顺利的进行，才能事半功倍。

如何成为一名全面的嵌入式工程师？

经验告诉我们最好掌握 3 类 MCU，第一类 51 系列，这是入门的基础，也是低成本设计的首选，毕竟没有不想省钱的老板，同时很多应用设计，好比要设计的是拖拉机而发动机却用的是波音 747 的引擎，结果自然就是一启动就四塴五裂，或者也只能是个毫无性价比的摆设。其次，AVR、PIC、430 等这类中档型的 MCU（微控制器）能掌握其中一种，毕竟有很多应用场合环境恶劣，或者追求低功耗需要电池供电方式等，这就对系统的性能有了更多要求。最后，学习和掌握 ARM，不仅仅入门难度和应用层次的提高，同样也是编程思想上的转变，8 位、16 位甚至 ARM7 系列 MCU 的编程更多的是逻辑性编程，是按照时序和外围设备进行通信，通过数据交换完成指令和数据的传递，实现智能控制应用设计。而到了 ARM9 以及更高性能的 MPU（微处理器）来说则是面向对象编程，在应用层基础上对底层驱动的调用和裁剪。

JASK2132 开发板的设计定位是什么？

首先，考虑到 ARM7 的研究和学习价值。ARM7 内核 MCU 的处理速度在 8 位机的基础上有了本质的提高，内部集成的 SPI、PWM、RTC、UART 等功能模块也更丰富更完善，解决了很多 8 位机应用的技术瓶颈，虽然 ARM7 远没有 ARM9 系列强大，但 ARM7 入门的门槛相比 AMR9 要低，而且开发工具和开发环境的调试方法一致，所以，在嵌入式 IC 大家族中，ARM7 往往在 8 位机向 32 位机应用转变过程中扮演着重要的桥梁

作用，起到平和过渡效果。

其次，一方面我们自身有必要采用 ARM7 芯片对主流无线芯片进行结合性应用评估，于此同时也可以为广大嵌入式应用提供了应用参考，降低开发风险，提高开发效率，加速项目进程。

再次，我们考虑到 8051 使用群体最广，所以设计中以 8051 作为药引，以 ARM 为核心重点，以无线作为纽带，对重点和难度做出详细分析，例程举一反三；并提供入门视频讲解，从简单到复杂，从入门到精通，学以致用，让嵌入式技术的学习和应用充满活力。我们也将不断更新和完善，希望我们理念也能引导、感悟和改变每一位热爱和向往嵌入式技术应用开发人员的现在和未来。

嵌入式无线通信技术的典型应用领域

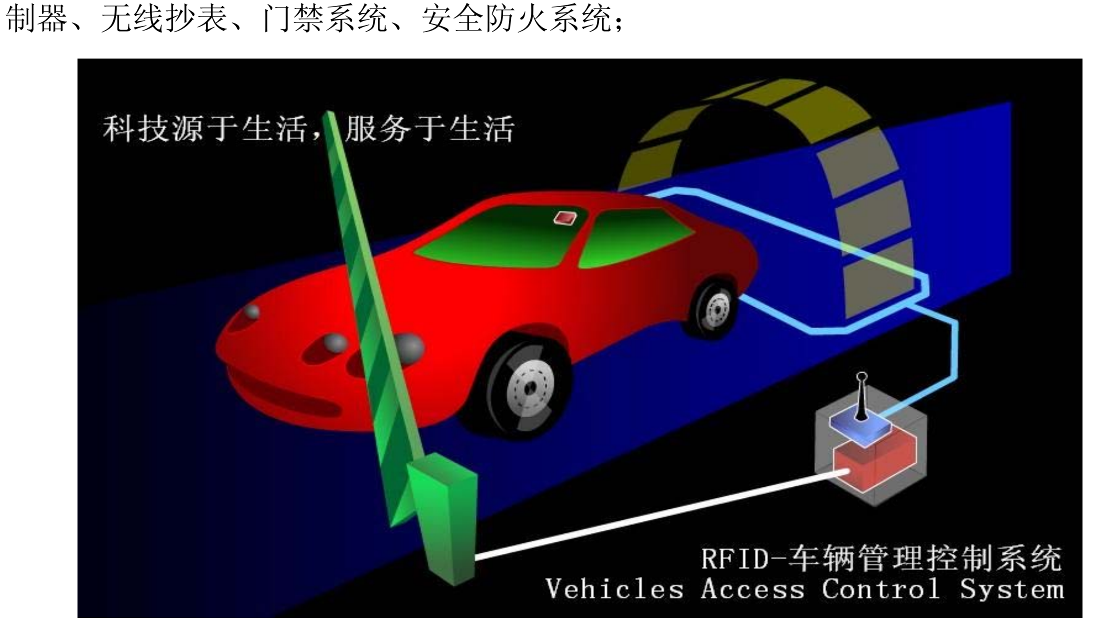
RFID 识别类：无线标签、身份识别、非接触 RF 智能卡

无线网络：无线局域网络，无线星形网络，无线拓扑网络

数据采集类：无线工业数据采集，无线 232/485/422 数据通信

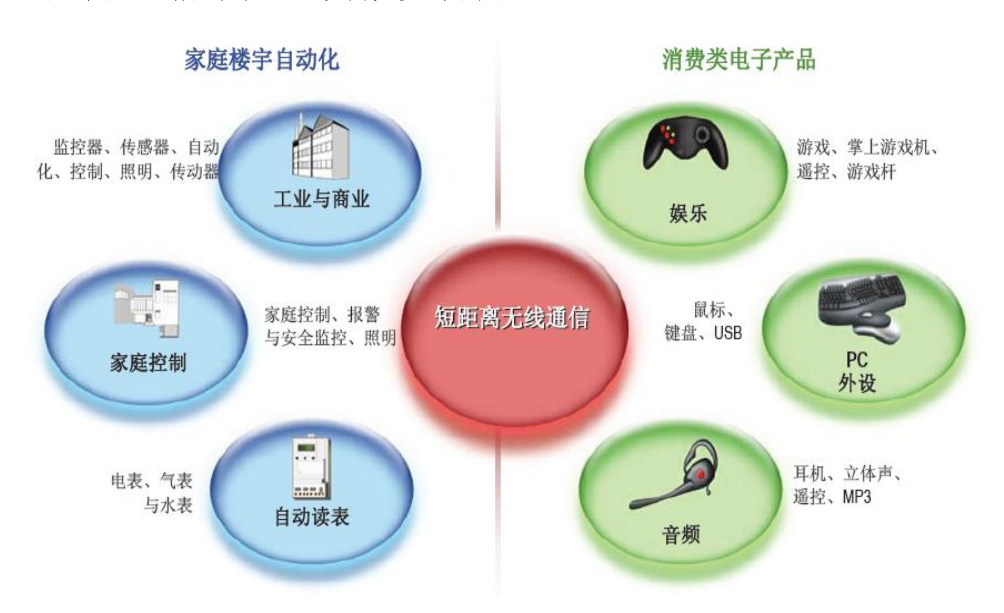
无线传感器：温度、压力等传感器信号无线采集、机器人控制

检测监控类：车辆管理系统、遥控引爆、工业遥控、无线鼠标键盘、遥测、航模控制器、无线抄表、门禁系统、安全防火系统；



第二部分 ARM7-LPC2132 芯片解析

ARM也许你现在还不太熟悉和精通，但相信每一位电子工程师都会对此有所兴趣，ARM即Advanced RISC Machines的缩写,既可以认为是一个公司的名字，也可以是对一类微处理器的通称，ARM架构是一款面向低预算市场设计的RISC微处理器，具有性能高、成本低和低能耗的特点，适用于嵌入控制、消费类多媒体、DSP和个人通信等嵌入式等众多领域。



同时，我们也已经真正进入一个无线技术无所不在的时代。手机通话、短信息通信无处不在；GPS 导航系统为我们导航指路；无线智能家居设备、无线故障监测系统、农作物环境监测控制系统等典型应用，而这一切，正是嵌入式技术和无线通通信技术相结合的产物，嵌入式无线数据通信技术的迅速发展，正不断改变人们生活的方方面面，让人们的生活更加舒适、美好和安全。

多年以来，我们一直致力于短距离无线通信技术的研究开发，积累了诸多经验和应用案例，短距离无线通信技术以方便、快捷的优点，必将成为诸多领域创新的

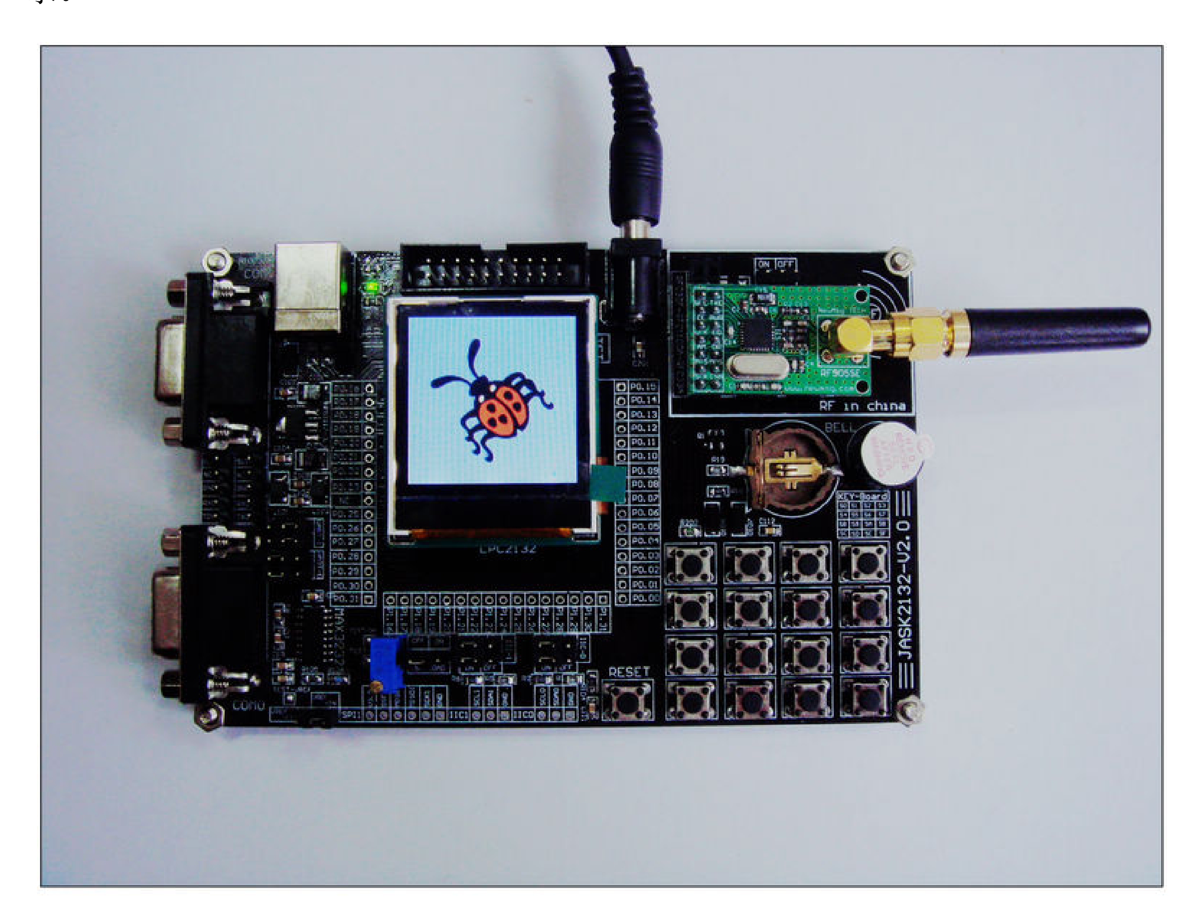
敲门砖，其中微处理器同样是无线通信技术发挥性能的心脏，是基础的基础，而目前MCU从8位到32位，型号成百上千，已是百花齐放。JASK2132正式是一款以PHILIPS公司出品的LPL2132（ARM7）芯片作为主控制器,并与主流无线芯片相结合的系列开发板，传统的8位单片机，随着网络、通信、多媒体时代的到来，也逐渐显现出了一些先天性的不足和设计瓶颈，而32位嵌入式系统在系统主频，运算处理能力上有了本质的提高，满足控制系统的网络化、智能化的发展趋势，为高端产品应用设计提供了广阔的空间。

LPC2132不是万能的，也不是最强大，甚至说是ARM系列中基础型的产品，但在嵌入式应用中，LPC系列以开发相对简单，资料丰富而广受欢迎，是ARM系列芯片大家族中的一个简约而不简单的明星。正是考虑到LPC系列的这些优点，将LPC2132与主流的NRF905、CC1101、NRF24L01等无线芯片结合起来，抛开传统呆板、空洞的评估方式，引领技术时尚，以形象的案例应用，让你拥有感性认识的同时，激发无限的创新能力。学以致用，方显学习之根本。研发JASK2132开发板的目的在于将主流的ARM和RF相结合起来，并做典型应用评估，让大家认识嵌入式，认识无线通信技术，让工程师掌握嵌入式开发技巧，让越来越多的人设计出各行各业迫切需要的行业应用。致力于嵌入式无线通信技术的推广和快速发展，是我们的宗旨所在。于此同时，我们也在不断开发更多ARM9、ARM11等系列产品。

LPC213X 概述

LPC2131/2132/2134/2136/2138 微控制器是基于一个支持实时仿真和嵌入式跟踪的ARM7TDMI-S内核的通用16/32位CPU微处理器，ARM7TDMI-S为冯-诺依曼结构，同时，采用3级流水线技术，处理和存储系统的所有部分都可以连续工作，具有高性能和低功耗的特性，对代码规模有严格控制的应用可使用16位Thumb模式将代码规模降低超过 30%，而性能的损失却很小，单电源供电模式，应用已经非常普及，已经广泛应用于个人通信等嵌入式领域，特别适用于工业控制应用以及医疗系统。

你会逐渐发现 ARM7都可以实现几乎所有的嵌入式应用,或采用定制的方式来满足需求。



LPC213X基本特性

01. 16/32 位 ARM7TDMI-S 核
02. 集成 8/16/32kB 的 RAM 和 32/64/128/256/512kB 的片内 Flash 程序存储器
03. 128 位宽度接口/加速器可实现高达 60MHz 工作频率
04. 可通过片内 boot 装载程序实现在系统编程/在应用编程(ISP/IAP)
05. 单个 Flash 扇区或整片擦除时间为 400ms，256 字节行编程时间为 1ms
06. 嵌入式跟踪接口可以对代码进行实时调试和高速跟踪
07. LPC2131/LPC21328 路 10 位的 A/D 转换器(LPC2134/36/38 为 16 路)
08. 1 个 10 位的 D/A 转换器，可产生不同的模拟输出

09. 2 个 32 位定时器/外部事件计数器(带 4 路捕获和 4 路比较通道)

10. PWM 单元(6 路输出)并集成片内看门狗功能

11. 低功耗实时时钟具有独立的电源和特定的 32kHz 时钟输入

12. 2 个 16C550 工业标准 UART0 和 UART1

13. 2 个高速I^2C总线(400 kbit/s)

14. SPI和具有缓冲作用和数据长度可变功能的SSP

15. 向量中断控制器。可配置优先级和向量地址

16. 47 个通用I/O口(I/O口可承受 5V的电压)

17. 9 个边沿或电平触发的外部中断管脚

18. 通过片内PLL(100us的设置时间)可实现最大为 60MHz的 CPU操作频率

19. 片内集成振荡器与外部晶体的操作频率范围为 1～30 MHz

20.与外部振荡器的操作频率范围高达 50MHz

21.低功耗模式：空闲和掉电

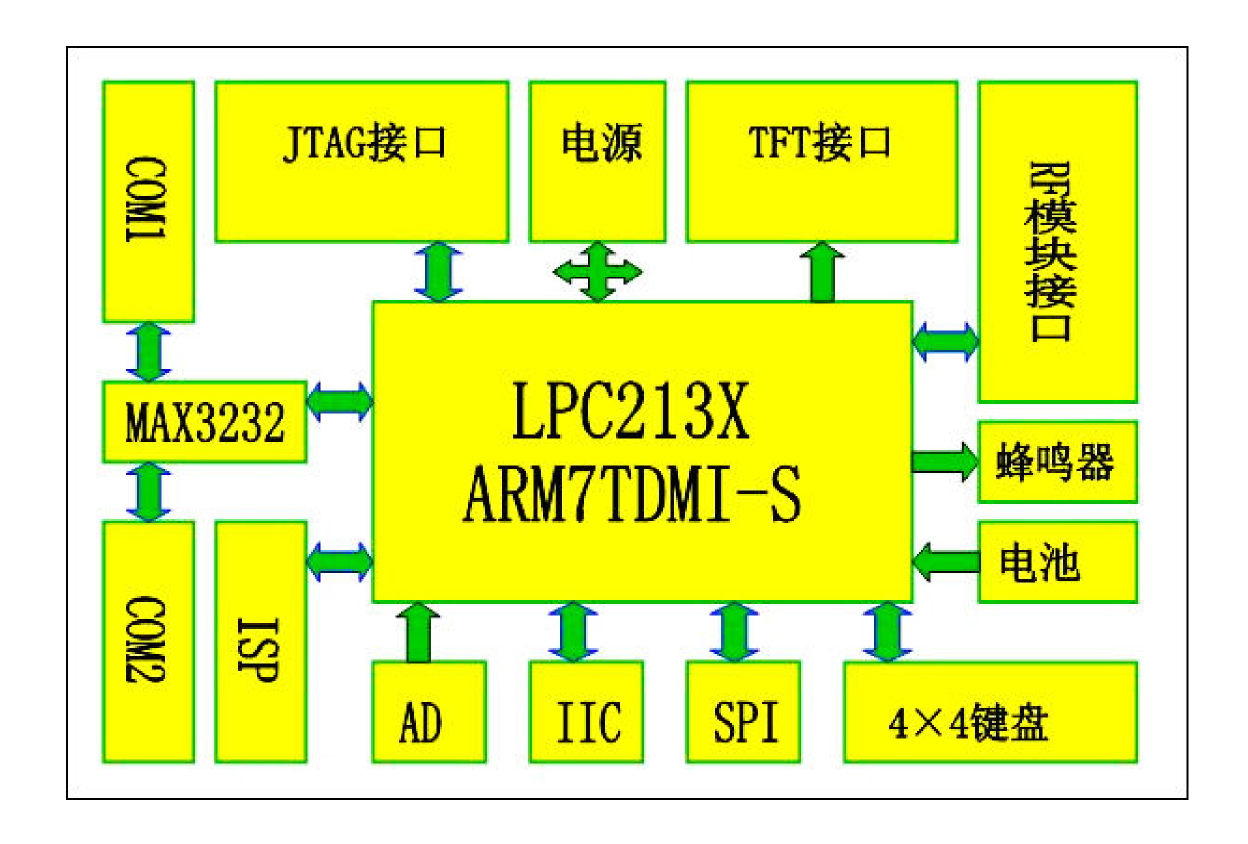
22.可通过个别使能/禁止外部功能和外围时钟分频来优化功耗

23.通过外部中断或BOD将处理器从掉电模式中唤醒

24.单电源，具有上电复位(POR)和掉电检测(BOD)电路

25.CPU操作电压范围：3.0~3.6 V (3.3 V±10%)

硬件功能示意图



锁相环 PLL 和 VPB 分频器要点解析

通过分清晶振频率(FOSC)、处理器时钟(Fcclk)、系统外设时钟(Fpclk)、CCO 时钟的意义，可以学习和掌握通过对锁相环 PLL 和 VPB 分频器的配置，灵活实现所想要的时钟系统。

锁相环 PLL 和 VPB 分频器概述和意义

PLL 锁相环（倍频）：由于输入时钟频率范围为 10～25MHz，频率不高，而外部晶体振荡器输出的时钟信号，通过 PLL 升频，可以获得更高的系统时钟（CCLK）。通过一个电流控制振荡器（CCO）倍增到 10～60MHz。

PLL 锁相环主要用途如下：

(1)可以通过 PLL 提升系统主频；

(2)增强抗干扰性能。

VPB 分频器（分频）：对 PLL 时钟分频，供给片上外设使用， 由于 CPU 内核的速度通常比外设部件的工作速度快，用户通过使用设置 VPB 分频器，将 Fcclk 信号降低到一个合适的值 Fpclk， 因此，VPB 分频器决定处理器时钟（CCLK）与外设器件所使用的时钟（PCLK）之间的关系。

VPB 分频器主要有两个用途：

(1)将 Fcclk 分频，为外设提供可在合适的速度下工作所需的 Fpclk 时钟；

(2)降低系统功耗（工作频率越高，功耗越大）。

四类时钟频率的区别和各自作用

(1)晶振频率(FOSC)：外部晶振的频率，如 JASK2132 上为 11.0592MHZ 晶振。

(2)处理器时钟(Fcclk)：芯片执行指令的频率。Fcclk = FOSC × PLL 倍频。对应的寄存器：PLLCFG[4:0]。

(3)VPB 时钟(Fpclk)：给片内外设提供的时钟频率。Fpclk=Fcclk/PLL 分频。对应的寄存器：APBDIV

(4)CCO 时钟：无实用的意义。只要把它设置在 156MHz~320MHz 范围内就行。对应的寄存器：PLLCFG[6:5]

FOSC的范围：10MHz~25MHz

Fcclk的范围：10MHz~60MHZ（最大允许频率60MHZ）

CCO的范围：156MHz~320MHz

锁相环 PLL 和 VPB 分频器基本操作方法

PLL 配置需要的寄存器如下：

PLLCON：PLL 控制寄存器。控制 PLL 使能（PLLCON=1）和 PLL 连接的状态。

PLLCFG：PLL 配置寄存器。设置 PLL 倍频（M 值）和 PLL 分频（P 值）。

PLLSTAT：PLL 状态寄存器。读出 PLL 状态（PLSTAT.10=11 等待 PLL 锁定。

PLLFEED：PLL 馈送寄存器。将值 0xAA、0x55 写入 PLLFEED，才能使 PLLCON（PLLCON=3）和 PLLCFG 的配置生效。

PLL 和 VPB 操作示例

```
/* 设置系统各部分时钟 */
PLLCON = 1;                          //  使能 PLL，但是不连接 PLL
#if (Fpclk / (Fcclk / 4)) == 1
VPBDIV = 0;
#endif
#if (Fpclk / (Fcclk / 4)) == 2
VPBDIV = 2;
#endif
#if (Fpclk / (Fcclk / 4)) == 4
VPBDIV = 1;
#endif
#if (Fcco / Fcclk) == 2
PLLCFG = ((Fcclk / Fosc) - 1) | (0 << 5);
#endif
#if (Fcco / Fcclk) == 4
PLLCFG = ((Fcclk / Fosc) - 1) | (1 << 5);
#endif
#if (Fcco / Fcclk) == 8
PLLCFG = ((Fcclk / Fosc) - 1) | (2 << 5);
#endif
#if (Fcco / Fcclk) == 16
PLLCFG = ((Fcclk / Fosc) - 1) | (3 << 5);
#endif
PLLFEED = 0xaa;                        //向 PLL 送序列
PLLFEED = 0x55;
while((PLLSTAT & (1 << 10)) == 0);     //等待 PLL 锁定
PLLCON = 3;                          //设置激活并连接 PLL
PLLFEED = 0xaa;
PLLFEED = 0x55;
```

在操作 PLL 和 VPB 时的注意事项

(1)PLL 在芯片复位和进入掉电模式时被关闭并旁路。PLL 只能通过软件使能；程序必须在配置并激活 PLL 后等待其锁定，然后再连接 PLL;

(2)PLL 值的不正确设定会导致芯片的错误操作;

(3)PLL在作为时钟源之前必须进行设置、使能并锁定。将振荡器时钟切换到PLL输出或反过来操作时，内部电路对操作进行同步以确保不会产生干扰。硬件不能确保PLL在连接之前锁定或在PLL在失去锁定时自动断开连接。在PLL失去锁定的情况下，振荡器很可能已经变得不稳定，这样断开PLL也挽救不了这种状况;

(4)仅当正确馈送序列写入PLLFEED寄存器才能使PLLCON和PLLCFG寄存器更改;

(5)掉电模式会自动关闭并断开 PLL。从掉电模式唤醒不会自动恢复 PLL 的设定，PLL 的恢复必须由软件来完成

备注：以上是对 PLL 和 VPB 功能的介绍、基本操作方法以及注意事项做出了理论性综述，请结合 JASK2132 进行实战操作，方能得到理性升华，切勿浮躁，详细说明请对照 LPC2132 文档对应寄存器仔细查阅。

中断操作要点解析

向量中断控制器（VIC）概述

向量中断控制器（VIC）具有32个中断请求输入，可将其编程分为3类：FIQ（快速中断请求）、向量IRQ和非向量IRQ。而可编程分配机制意味着不同外设的中断优先级可以动态分配并调整。

快速中断请求（FIQ）要求具有最高优先级。如果分配给FIQ的请求多于1个，VIC将中断请求“相或”后向ARM处理器产生FIQ信号。当只有一个中断被分配为FIQ时可实现最短的FIQ等待时间，因为FIQ服务程序只要简单地启动器件的处理就可以了。

但如果分配给FIQ级的中断多于1个，FIQ服务程序从VIC中读出一个字来识别产生中断请求的FIQ中断源是哪一个。

向量IRQ具有中等优先级。该级别可分配32个请求中的16个。32个请求中的任意一个都可分配到16个向量IRQ slot中的任意一个，其中slot0具有最高优先级，而slot15则为最低优先级。

非向量IRQ的优先级最低。

VIC将所有向量和非向量IRQ“相或”向ARM处理器产生IRQ信号。IRQ服务程序可通过读取VIC的一个寄存器立即启动并跳转到相应地址。如果有任意一个向量IRQ发出请求，VIC则提供最高优先级请求IRQ服务程序的地址，否则提供所默认程序的地址。该默认程序由所有非向量IRQ共用。默认程序可读取另一个VIC寄存器以确定哪个IRQ被激活。

向量中断控制器（VIC）特性

(1)ARM PrimeCell ™向量中断控制器

(2)最多32个中断请求输入

(3)16个向量IRQ中断

(4)16个优先级，可动态分配给中断请求

(5)可软件中断产生

(6)VIC中所有的寄存器都为字寄存器。不支持字节和半字的读和写操作。

VIC 基本操作方法

VIC 的基本操作方法如下：

(1)设置中断是 FIQ 还是 IRQ，若是 IRQ，再设置向量中断，并分配中断优先级。

(2)设置中断允许，以及乡里向量中断对应地址或非向量中断默认地址。

(3)有中断之后，若是 IRQ 中断，可读取向量地址寄存器，然后跳转到服务程序。

(4)当要退出中断时，对向量地址寄存器写 0，通知 VIC 中断结束

中断操作示例

```
//以定时器 0 中断为例子
IRQEnable();                        /* IRQ 中断使能

/* 设置定时器 0 中断 IRQ */
VICIntSelect = 0x00;                /* 所有中断通道设置为 IRQ 中断
        */
VICVectCntl0 = 0x20 | 0x04;          /* 设置定时器 0 中断通道分配最高优
先*/
VICVectAddr0 = (uint32)IRQ_Timer0;  /* 设置中断服务程序地址
    */
VICIntEnable = 1 << 0x04;            /* 使能定时器 0 中断

/*对应中断服务程序*/
void __irq IRQ_Timer0 (void)
{
    if ((IO0SET & BEEP) == 0)
    IO0SET = BEEP;                  /* 关闭BEEP*/
    else
    IO0CLR = BEEP;
    T0IR = 0x01;                    /* 清除中断标志*/
    VICVectAddr = 0x00;              /* 通知VIC中断处理结束*/
}
```

VIC 使用注意事项

如果在片内 RAM 当中运行代码并且应用程序需要调用中断,那么必须将中断向量重新映射到 Flash 地址 0x0。这样做是因为所有的异常向量都位于地址 0x0 及以上。通过将寄存器 MEMMAP(位于系统控制模块当中)配置为用户 RAM 模式来实现这一点。用户代码被连接以便使中断向量表 （IVT）装载到 0x4000 0000。虽然可以选择多个中断源（通过 VICIntSelect）来产生 FIQ 请求，但是只有一个专门的中断服务程序来服务响应所有可用/出现的 FIQ 请求。因此，如果分配为 FIQ 的中断多于一个，FIQ 中断服务程序就必须读取 VICFIQStatus 的内容来决定如何处理中断请求。不过我们还是建议只将一个中断分配为 FIQ。多个 FIQ 中断源会增加中断延迟。

在中断服务程序执行完毕后，对外设中断标志的清零将会对VIC寄存器

（VICRawlntr，VICFIQStatus和VICIRQStatus）当中的对应位产生影响。另外，为了能够服务下次中断，必须在中断返回之前对VICVectAddr寄存器执行写操作。该写操作将清零内部中断优先级硬件当中对应的中断标志。

通常要禁止VIC中断，必须清零VICIntEnClr寄存器中的对应位，该操作使VICIntEnable寄存器的对应位清零。这同样应用于VICSoftInt和VICSoftIntClear，VICSoftIntClear将会使VICSoftInt中的对应位清零。例如，如果VICSoftInt=0x0000 0005并且bit0必须清零，那么VICSoftIntClear=0x0000 0001可实现该操作。通过写VICSoftIntClear来执行对VICSoftInt当中相同位的新的清零操作之前，必须执行VICSoftIntClear=0x0000 0000。因此向VICSoftIntClear寄存器任何位写入1对目标寄存器都是一次有效。

如果看门狗只在溢出或无效喂狗时产生中断，那么无法清除中断。唯一的方法是通过VICIntEnClr禁止VIC中断来实现中断返回。

假设UART0和SPI0产生中断请求，它们被分配为向量IRQ（UART0的优先级高于SPI0），而UART1和I^{2}C产生非向量IRQ，下面就是VIC一种可能的设定：

VICIntSelect = 0x0000 0000
//(SPI0，I^{2}C，UART1和UART0为IRQ => bit10，bit9，bit7和bit6=0)
VICIntEnable = 0x0000 06C0
//(SPI0，I^{2}C，UART1和UART0中断使能 => bit10，bit9，bit 7和bit6=1)
VICDefVectAddr = 0x…
//(保存服务非向量IRQ的程序地址（即，UART1和I^{2}C的起始地址))
VICVectAddr0 = 0x… //(保存UART0 IRQ服务程序的起始地址)
VICVectAddr1 = 0x…
//(保存SPI0 IRQ服务程序的起始地址)
VICVectCntl0 = 0x0000 0026
//(VIC通道号为6（UART0）的中断源使能为优先级0（最高级）)
VICVectCntl1 = 0x0000 002A
//(VIC通道号为10（SPI0）中断源使能为优先级1)

在任何IRQ请求（SPI0，I^{2}C，UART0或UART1）产生之后，微控制器跳转到地址0x00000018执行代码。对于向量和非向量IRQ，可在地址0x18放入下面指令：

```
LDR pc, [ pc, #-0xFF0 ]
```

该指令将VICVectAddr寄存器中保存的地址装入PC。

一旦产生UART0请求，VICVectAddr和VICVectAddr0相同。如果产生SPI请求，VICVectAddr等于VICVectAddr1。如果UART0和SPI都没有产生IRQ请求，而UART1和/或I^2C产生请求，那么VICVectAddr的内容与VICDefVectAddr相同。

备注：以上是对中断操作方法的介绍以及注意事项做出了理论性综述，请结合JASK2132进行实战操作，从模仿做起，举一反三，方可成本自己身本领，详细说明请对照LPC2132文档对应寄存器仔细查阅，

GPIO 操作要点解析

GPIO 功能概述

LPC213X系列芯片的引脚一般都是多功能复用的，可以通过引脚连接模块在多个功能之间进行选择,引脚连接模块配置寄存器控制多路开关来连接管脚与片内外设。外设在激活和任何相关中断使能之前必须连接到适当的管脚。任何使能的外设功能如果没有映射到相关的管脚，则被认为是无效的。当管脚只选择一个功能时，其它复用功能无效。系统复位时，所有通用GPIO默认为第一功能，其中引脚功能选择寄存器PINSEL0、PINSEL1、PINSEL2如下。

PINSEL0	管脚名称	00	01	10	11	复位值
1:0	P0.0	P0.0	TxD0	PWM1	保留	00
3:2	P0.1	P0.1	RxD0	PWM3	EINT0	00
5:4	P0.2	P0.2	SCL0	CAP0.0	保留	00
7:6	P0.3	P0.3	SDA0	MAT0.0	EINT1	00
9:8	P0.4	P0.4	SCK0	CAP0.1	AD0.6	00
11:10	P0.5	P0.5	MISO0	MAT0.1	AD0.7	00
13:12	P0.6	P0.6	MOSI0	CAP0.2	保留	00
15:14	P0.7	P0.7	SSEL0	PWM2	EINT2	00
17:16	P0.8	P0.8	TxD1	PWM4	保留	00
19:18	P0.9	P0.9	RxD1	PWM6	EINT3	00

PINSEL0	管脚名称	00	01	10	11	复位值
21:20	P0.10	P0.10	保留	CAP1.0	保留	00
23:22	P0.11	P0.11	保留	CAP1.1	SCL1	00
25:24	P0.12	P0.12	保留	MAT1.0	保留	00
27:26	P0.13	P0.13	保留	MAT1.1	保留	00
29:28	P0.14	P0.14	保留	EINT1	SDA1	00
31:30	P0.15	P0.15	保留	EINT2	保留	00

PINSEL1	管脚名称	00	01	10	11	复位值
1:0	P0.16	P0.16	EINT0	MAT0.2	CAP0.2	00
3:2	P0.17	P0.17	CAP1.2	SCK1	MAT1.2	00
5:4	P0.18	P0.18	CAP1.3	MISO1	MAT1.3	00
7:6	P0.19	P0.19	MAT1.2	MOSI1	CAP1.2	00
9:8	P0.20	P0.20	MAT1.3	SSEL1	EINT3	00
11:10	P0.21	P0.21	PWM5	保留	CAP1.3	00
13:12	P0.22	P0.22	保留	CAP0.0	MAT0.0	00
15:14	P0.23	P0.23	保留	保留	保留	00
17:16	P0.24	保留				00
19:18	P0.25	P0.25	AD0.4	保留	保留	00
21:20	P0.26	保留	AD0.5	保留	保留	00
23:22	P0.27	P0.27	AD0.0	CAP0.1	MAT0.1	00
25:24	P0.28	P0.28	AD0.1	CAP0.2	MAT0.2	00
27:26	P0.29	P0.29	AD0.2	CAP0.3	MAT0.3	00
29:28	P0.30	P0.30	AD0.3	EINT3	CAP0.0	00
31:30	P0.31	P0.31	保留	保留	保留	00

PINSEL2	描述	复位值
1:0	保留，用户软件不要向其写入 1。	00
2	该位为 0 时，P1.36:26 用作 GPIO。该位为 1 时，P1.31:26 用作一个调试端口。	P1.26/RTCK
3	该位为 0 时， P1:25:16 用作 GPIO。 该位为 1 时， P1.25:16 用作一个跟踪端口。	P1.20 / TRACESYNC
31:28	保留	NA

4. 管脚功能选择寄存器值

PINSEL 寄存器控制下表中器件管脚的功能。每一对寄存器位对应一个特定的器件管脚。

管脚功能选择寄存器位

PINSEL0 和 PINSLE1 的值		功能	复位值
0	0	首选（默认）功能，通常为 GPIO 口	00
0	1	第一可选功能	
1	0	第二可选功能	
1	1	第三可选功能	

只有当管脚选择 GPIO 功能时，IO0DIR/IO1DIR 寄存器的方向控制位才有效。其它功能的方向是自动控制的。每个派生器件通常具有不同的管脚分布，因此每个管脚可能有不同的功能。

功能模块的操作方法

以设置 P0.0、P0.1 为 TXD0,RXD0 为例

PINSEL0 = 0x00000005;   // 设置 I/O 连接到 UART0

LPC2132 拥有多达 47 个通用 GPIO，分别是 P[31:0]和 P[31:16]，其中，P0.24 未用，P0.31 仅能作为输出端口，由于所有 GPIO 都有多种功能复用，所以首先需要通过 PINSEL0、PINSEL1、PINSEL2 来选择和确定功能，然后通过 IODIR 寄存器设置来确定 GPIO 的方向（输入/输出），然后就可以通过 IOSET、IOCLR 和 IOPIN 这 3 个寄存器来实现对 IO 的置位、清零和读取 IO 状态。

通用名称	描述	访问	复位值	PORT0 地址&名称	PORT1 地址&名称
IOPIN	**GPIO 管脚值寄存器** 不管方向和模式如何设定，管脚的当前状态都可从该寄存器中读出。	只读	NA	0xE0028000 IO0PIN	0xE0028010 IO1PIN
IOSET	**GPIO 输出置位寄存器** 该寄存器和 IOCLR 寄存器一起控制输出管脚的状态，写入 1 使对应管脚输出高电平，写入 0 无效。	读/写	0x00000000	0xE0028004 IO0SET	0xE0028014 IO1SET
IODIR	**GPIO 方向控制寄存器** 该寄存器单独控制每个 I/O 口的方向。	读/写	0x00000000	0xE0028008 IO0DIR	0xE0028018 IO1DIR
IOCLR	**GPIO 输出清零寄存器** 该寄存器控制输出管脚的状态，写入 1 使对应管脚输出低电平并清零 IOSET 寄存器中的对应位，写入 0 无效。	只写	0x00000000	0xE002800C IO0CLR	0xE002801C IO1CLR

GPIO 的读、写操作方法

以下以 GPIO-0.21 示例

```
IO0SET = 1 << 21;     //对 P0.21 端口置 1
IO0CLR = 1 << 21;     //对 P0.21 端口清 0
IO0DIR|= 1 << 21;   //设置 P0.21 方向为输出
```

如果指定输出管脚在GPIO输出置位寄存器（IOnSET）和GPIO输出清零寄存器（IOnCLR）中的对应位都置位，那么管脚的输出电平取决于后写入的寄存器的值。例如：

```
IO0SET = 0x0000 0080
IO0CLR = 0x0000 0080
```

P0.7输出电平为低，因为写GPIO清零寄存器在写置位寄存器之后。

如果在应用中要求在特定的并行口上瞬时出现0和1，可直接通过访问相应的GPIO管脚值寄存器（IOPIN）来实现。

假设P0.8～P0.15要配置成输出，则写IO0PIN：

IO0PIN=0x0000 C700

产生的输出结果与下面两条写指令的相同：

IO0SET=0x0000 C700

IO0CLR=0x0000 3800

由此可见，前者较后者更简便

GPIO 使用注意事项

以下以设置I/O连接到UART0为例

```
PINSEL0=0x00000005;
PINSEL0=(PINSEL0&(0xfffffff0))|0x00000005;
```

前者是直接对PINSEL0进行赋值，而后者是在保证其他引脚功能不变的情况下对P0.0和P0.1设置为TXD0，RXD0功能。这样的优点是，便于代码的随便调用。举一反三，请思考以下的设置所实现的功能。

```
PINSEL1=PINSEL1&0x30000000;//设置TFT对应GPIO为数字IO功能
IO0DIR=IO0DIR|0xBFFF0000; //P0.16-0.23 ,p0.25-29,P0.31 都设为输出
```

总结：本节主要描述了 GPIO 的基本操作方法，GPIO 是嵌入式应用的基础，任何外围设备都是用 GPIO 来连接和驱动的，所以 GPIO 的操作是完成嵌入式应用设计的关键。请多加练习，习惯就成自然，伟大程序员就是这么练成的。

UART 应用要点解析

UART 功能概述

LPC2132 拥有 2 个符合’550 工业标准的异步串行口 UART0 和 UART1，者仅仅是外设地址不同，其他操作方法等完全一样。

UART 基本特性：

16 字节收发 FIFO；

寄存器位置符合’550工业标准；

接收器FIFO触发点可为1，4，8和14字节；

内置波特率发生器；

包含使能实现软件流量控制机制。

UART 管脚描述

管脚名称	UART 管脚	功能描述	说明
P0.0	TxD0	串行输出	串行发送数据
P0.1	RxD0	串行输入	串行接收数据
P0.8	TxD1	串行输出	串行发送数据
P0.9	RxD1	串行输入	串行接收数据

名称	描述	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0	访问	复位值*	地址
U0RBR	接收缓冲	MSB			读数据				LSB	RO	未定义	0xE000C000 DLAB=0
U0THR	发送保持	MSB			写数据				LSB	WO	NA	0xE000C000 DLAB=0
U0IER	中断使能	0	0	0	0	0	使能Rx线状态中断	使能THRE中断	使能Rx数据可用中断	R/W	0	0xE000C004 DLAB=0
U0IIR	中断ID	FIFO 使能		0	0	IIR3	IIR2	IIR1	IIR0	RO	0x01	0xE000C008
U0FCR	FIFO 控制	Rx 触发		保留		-	Tx FIFO 复位	Rx FIFO 复位	FIFO 使能	WO	0	0xE000C008
U0LCR	线控制	DLAB	设置间隔	奇偶固定	偶选择	奇偶使能	停止位个数	字长度选择		R/W	0	0xE000C00C
U0LSR	线状态	Rx FIFO 错误	TEMT	THRE	BI	FE	PE	OE	DR	RO	0x60	0xE000C014
U0SCR	高速缓存	MSB							LSB	R/W	0	0xE000C01C
U0DLL	除数锁存 LSB	MSB							LSB	R/W	0x01	0xE000C000 DLAB=1
U0DLM	除数锁存 MSB	MSB							LSB	R/W	0	0xE000C004 DLAB=1

UART中最重要的参数就是波特率概念，LPC2132中UART的波特率计算方法如下：

UART除数锁存LSB寄存器—UART Divisor Latch LSB Register
（U0DLL - 0xE000C000，U1DLL - 0xE0010000）
UART除数锁存MSB寄存器—UART Divisor Latch MSB Register
（U0DLM - 0xE000C004，U1DLM - 0xE0010004）

除数锁存是波特率发生器的一部分，它保存了用于产生波特率时钟的 VPB 时钟（*pclk*）分频值，波特率时钟必须是波特率的 16 倍，等式如下：

$$16 \times baud = \frac{Fpclk}{UxDLM, UxDLL}$$

UxDLL 和 UxDLM 寄存器一起构成一个 16 位除数，UxDLL 包含除数的低 8 位，UxDLM 包含除数的高 8 位。值 0x0000 被看作是 0x0001，因为除数是不允许为 0 的。

当访问 UART 除数锁存寄存器时，除数锁存访问位（DLAB）必须为 1。

UART 除数锁存 LSB 寄存器

UxDLL	功能	描 述	访问条件	访问	复位值
7:0	除数锁存 LSB 寄存器	UART 除数锁存 LSB 寄存器与 UxDLM 寄存器一起决定 UART 的波特率。	DLAB=1	读写	0x01

UART 除数锁存 MSB 寄存器

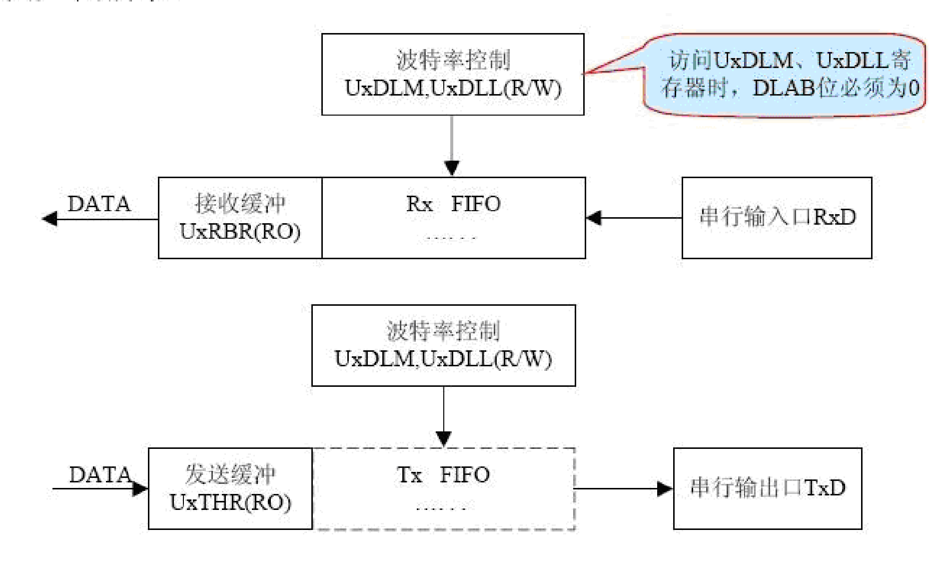
UxDLM	功能	描 述	访问条件	访问	复位值
7:0	除数锁存 MSB 寄存器	UART 除数锁存 MSB 寄存器与 UxDLL 寄存器一起决定 UART 的波特率。	DLAB=1	读写	0

而UART收发数据的数据格式又如何操作请看ULCR寄存器设置，如下：

UxLCR	功能	描述	复位值
1:0	字长度选择	00：5 位字符长度 01：6 位字符长度 10：7 位字符长度 11：8 位字符长度	0
2	停止位选择	0：1 个停止位 1：2 个停止位（如果 UxLCR[1:0]=00 则为 1.5）	0
3	奇偶使能	0：禁止奇偶产生和校验 1：使能奇偶产生和校验	0
5:4	奇偶选择	00：奇数 01：偶数 10：强制为 1 11：强制为 0	0
6	间隔控制	0：禁止间隔发送 1：使能间隔发送 当 UxLCR6=1 时，输出管脚 UART TxD 强制为逻辑 0。	0
7	除数锁存访问位	0：禁止访问除数锁存 1：使能访问除数锁存	0

UART 基本操作方法

两个串口具有完全相同的寄存器，只是物理地址不一样，UART的基本寄存器功能框图如 图 所示。



UART 的基本寄存器功能框图

设置GPIO连接到UARTx功能；

设置串口波特率(相关寄存器为UxDLM,UxDLL)；

设置串口工作模式(UxLCR,UxFCR)；

发送或接收数据(UxTHR, UxRBR)；

检查串口状态字(UxLSR)或者等待串口中断（UxIIR）

UART0 示例分析

```
/****************************************************************
** 函数名称 ：UART0_Init()
** 函数功能 ：串口初始化，设置工作模式和波特率。
** 入口参数 ：baud    波特率
**  set 模式设置(UARTMODE数据结构)
** 出口参数 ：1-初始化成功，  0-初始化失败
****************************************************************/
```

```
int8 UART0_Init (uint32 baud)
{
    uint32 bak;
/***************************参数过滤**********************************/
    if ((baud ==0 ) || (baud > 115200))    return (0);
    if ((set_datab <5) || (set_datab > 8)) return (0);
    if ((set_stopb == 0) || (set_stopb > 2))  return (0);
    if (set_parity > 4)  return (0);
/*******************************设置串口波特率 **********************/
    U0LCR = 0x80;                       // DLAB = 1
    bak   = (Fpclk >> 4) / baud;
    U0DLM = bak >> 8;
    U0DLL = bak & 0xFF;
/*******************************设置串口模式***************************/
    bak   = set_datab - 5;              // 设置字长
    if (set_stopb == 2)  bak |= 0x04;   // 判断是否为2位停止位

    if (set_parity != 0)
    {
        set_parity = set_parity - 1;
        bak |= 0x08;
    }
    bak |= set_parity << 4;             // 设置奇偶校验
    U0LCR = bak;
    return (1);
}
/**********************************************************************
** 函数名称 ： IRQ_UART0()
** 函数功能 ： 串口0接收中断服务程序
**********************************************************************/
void __irq IRQ_UART0 (void)
{
    uint8 i;
    if ((U0IIR & 0x0F) == 0x04)
        rcv_new = 1;          // 设置接收到新的数据标志
    for (i=0; i<4; i++)
    {
        rcv_buf[i] = U0RBR;      // 读取FIFO的数据，并清除中断
    }
    VICVectAddr = 0x00;          // 中断处理结束
}
```

```
/*********************************************************************
** 函数名称 ：UART0_SendByte()
** 函数功能 ：向串口0发送1字节数据
** 入口参数 ：dat 要发送的数据
*********************************************************************/
void UART0_SendByte (uint8 dat)
{
    U0THR = dat;  // 要发送的数据
}
    PINSEL0 = 0x00000005;              // 设置I/O连接到UART0
/********************定义串口模式设置数据结构 **********************/
    set_datab  = 8;       // 字长度选择，5位/6位/7位/8位可选
    set_stopb  = 1;       // 停止位，1/2可选
    set_parity = 0;       // 奇偶校验位，0-无校验,1-奇校验,2-偶校验
    rcv_new    = 0;
    UART0_Init(9600);                  // 串口初始化, 设置波特率等
// 如果你要设置成其他波特率如4800,只要把9600改成4800就可以啦,就这么简单,
    U0FCR = 0x41;                      // 使能FIFO，并设置触发点为4字节，也就
//是收到多少字节后才产生1次中断，PLC2132支持1、4、8、14字节中断模式
    U0IER = 0x01;                      // 允许RBR中断, 即接收中断

    IRQEnable();                       // 使能IRQ中断
/************************使能UART0中断 ******************************/
    VICIntSelect = 0x00000000;         // 设置所有的通道为IRQ中断
    VICVectCntl0 = 0x20 | 0x06;           // UART0分配到IRQ slot0，即最高优
先级
    VICVectAddr0 = (uint32)IRQ_UART0;  // 设置UART0向量地址
    VICIntEnable = 1 << 0x06;          // 使能 UART0 中断
```

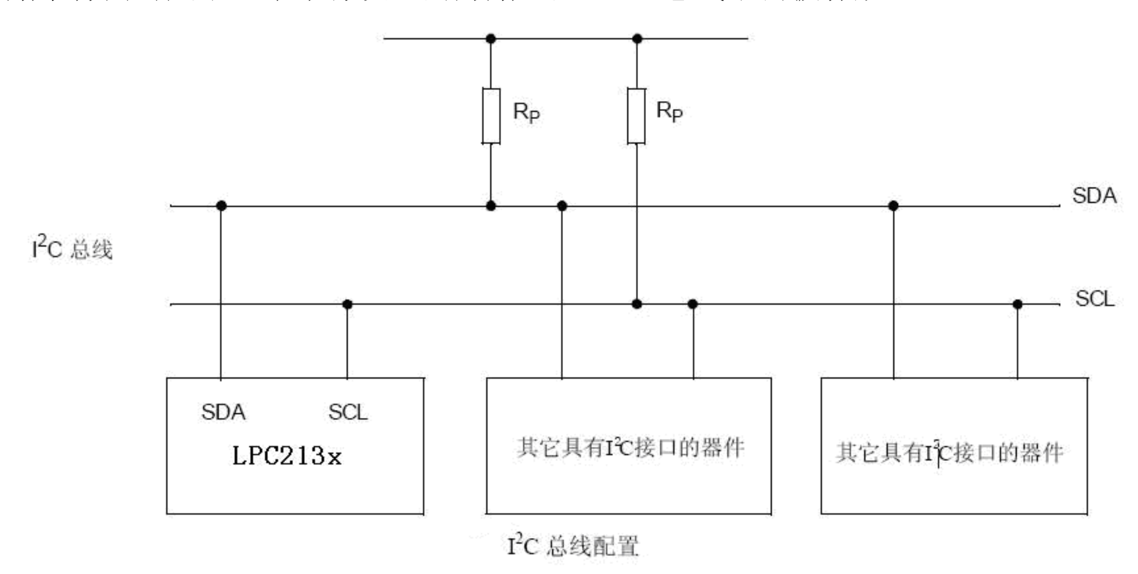
I2C 要点解析

I2C 功能概述

I2C总线的典型配置如图24所示。根据方向位(R/W)状态的不同，I2C总线上存在以下两种类型的数据传输：

(1)从主发送器向从接收器发送数据。主机发送的第一个字节是从机地址。接下来是数据字节流。从机每接收一个字节返回一个应答位。

(2)从发送器向主接收器发送数据。第一个字节(从地址)由主机发送。从机返回一个应答位。接下来从机向主机发送数据字节。主机每接收一个字节返回一个应答位。接收完最后一个字节，主机返回一个非应答位。主器件产生所有串行时钟脉冲和起始以及停止条件。出现停止条件或重复的起始条件时传输结束。由于重复的起始条件同时是下一个串行发送的开始，因此I2C总线不会被释放。



I2C 总线配置

I2C 基本特性

标准的 I2C 总线接口

可配置为主机、从机或主/从机

可编程时钟可实现通用速率控制

主机从机之间双向数据传输

多主机总线(无中央主机)

同时发送的主机之间进行仲裁，避免了总线数据的冲突

串行时钟同步使器件在一条串行总线上实现不同位速率的通信

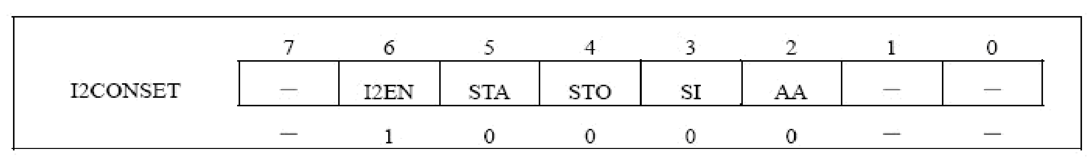
串行时钟同步可作为握手机制使串行传输挂起和恢复

I2C 总线可用于测试和诊断

I2C 操作模式

主发送器模式：

在该模式中，数据从主机发送到从机。在进入主发送器模式之前，I2C的ONSET必须按照图25进行初始化。必须置位I2EN来使能I2C功能。如果AA位为0，而另一个器件成为总线的主控器时，I2C将不会对任何地址产生应答。也就是说它无法进入从模式。STA，STO和SI必须设置为0。向I2CONCLR寄存器中的SIC位写入1可清零SI。



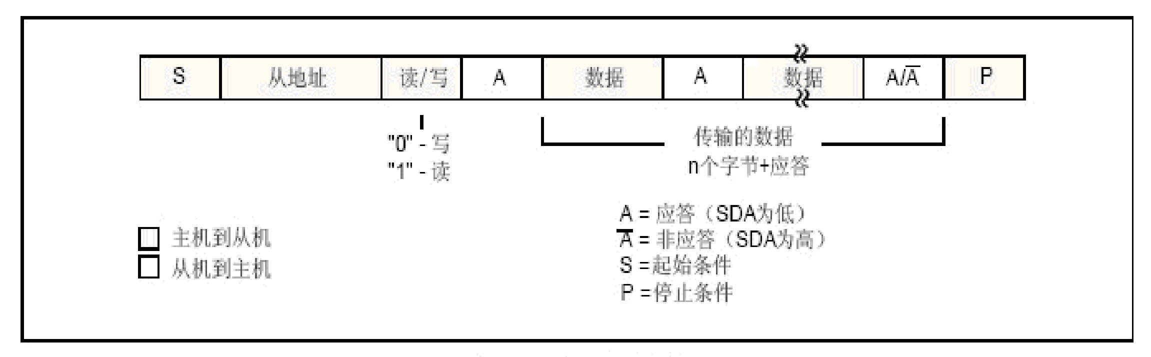
	7	6	5	4	3	2	1	0
I2CONSET	—	I2EN	STA	STO	SI	AA	—	—
	—	1	0	0	0	0	—	—

主模式配置

第一个发送的数据包含接收器件的从地址（7 位）和数据方向位。在此模式下，数据方向位（R/W）应当为 0 表示执行写操作。因此第一个发送的字节为从地址和写方向位。数据的发送每次为 8 位。每发送完一个字节，都接收到一个应答位。起始和停止条件用于指示串行传输的起始和结束。

通过软件置位STA进入I2C主发送器模式。I2C逻辑在总线空闲后立即发送一个起始条件。当发送完起始条件后，SI置位。此时I2STAT中的状态代码应当为08H。该状态代码用于指向一个中断服务程序。该中断程序将从地址和写方向位装入I2DAT（数据寄存器），然后清零SI位。向I2CONCLR寄存器中的SIC位写入1可清零SI。

当从地址和R/W位已发送且接收到应答位之后，SI位再次置位，并且对于主模式，可能的状态代码为18H，20H或38H，如果从模式使能（AA=1），可能的状态代码为68H，78H或0B0H。

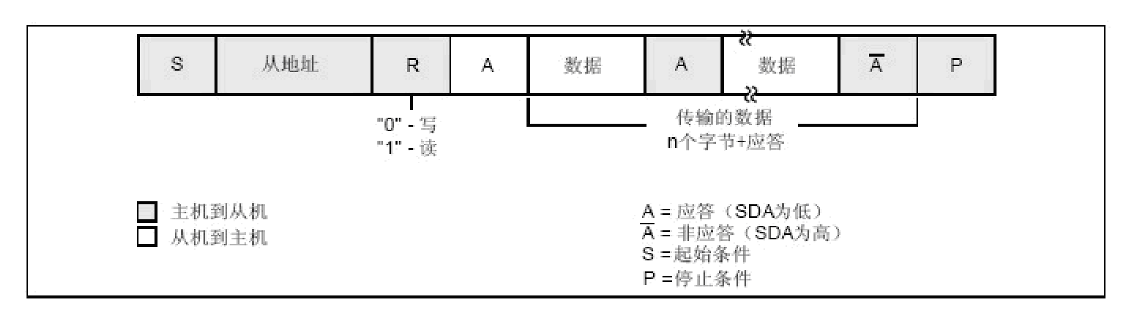


主发送器模式中的格式

主接收器模式

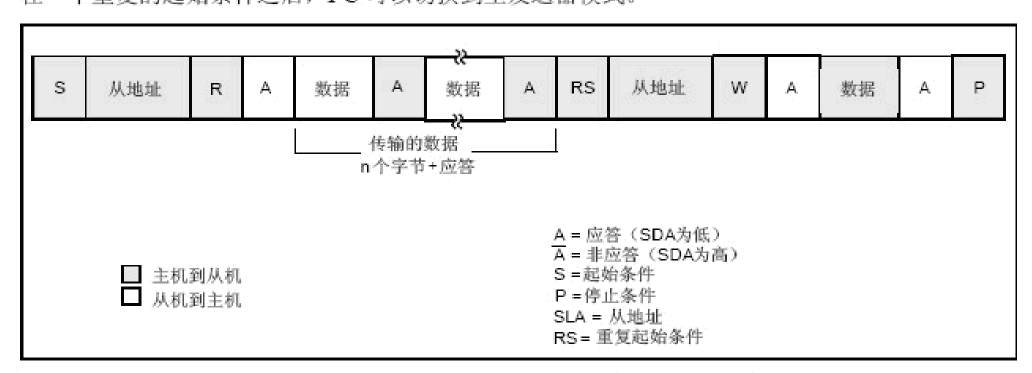
在主接收器模式中，数据字节接收自从发送器。传输的初始化与主发送器模式相同。发送完起始条件后，中断服务程序必须将从地址和数据方向位装入I2C数据寄存器（I2DAT），然后清零SI位。

当从地址和方向位已发送且接收到应答位之后，SI置位而状态寄存器将显示状态代码。对于主模式，可能的状态代码为40H，48H或38H，对于从模式，可能的状态代码为68H，78H或B0H。



主接收器模式的格式

在一个重复的起始条件之后，I²C 可以切换到主发送器模式。



在发送重复起始条件后，主接收器切换到主发送器

从接收器模式

在从接收器模式中，从主发送器接收数据字节。要初始化从接收器模式，用户必须将从地址写入从地 址寄存器（I2ADR）并配置I2C控制置位寄存器（I2CONSET）：

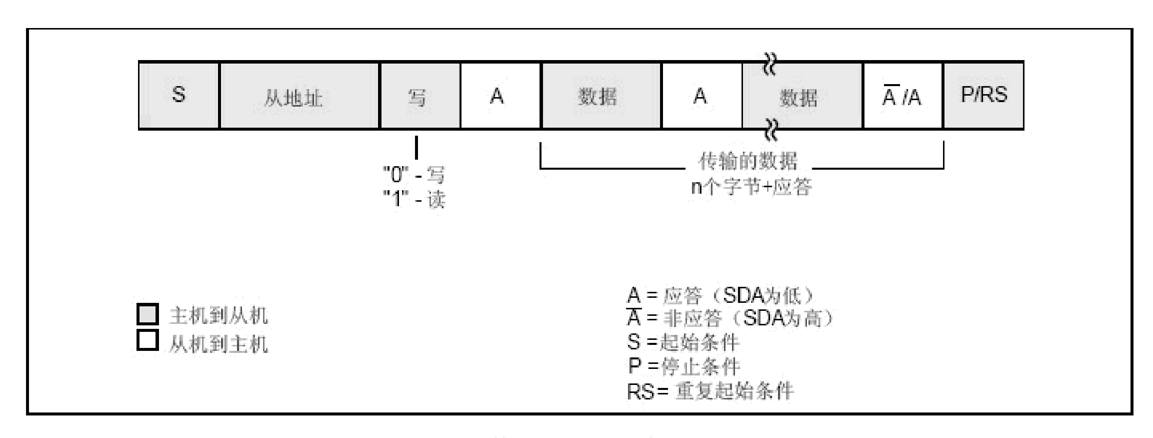


I2CONSET	7	6	5	4	3	2	1	0
	—	I2EN	STA	STO	SI	AA	—	—
	—	1	0	0	0	1	—	—

从模式配置

I2EN 必须置位以使能 功能。AA 位必须置位以使 I2C 应答自身的从地址或通用调用地址。STA，STO 和 SI 设置为 0。

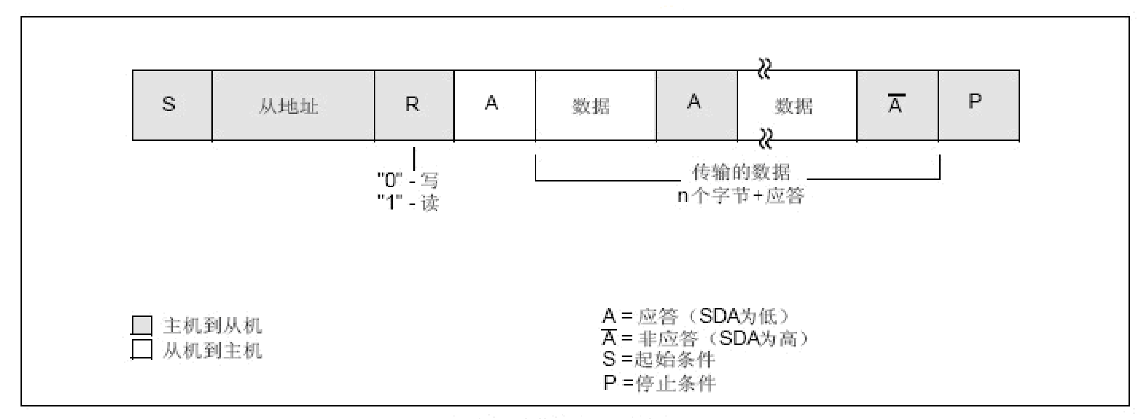
当I2ADR和I2CONSET完成初始化时，I^2C一直等待到它被自身的从地址或通用地址（两者后面都紧跟数据方向位）寻址为止。如果数据方向位为1(R)，I2C将进入从发送器模式。在接收到地址和方向位后，SI置位并可从状态寄存器（I2STAT）中读出有效的状态代码。



从接收器模式的格式

从发送器模式

第一个字节的接收和处理与从接收器模式相同。但在该模式中，方向位指示传输的方向掉转。串行数据通过SDA发送而串行时钟通过SCL输入。在串行传输的开始和结束对起始和停止条件进行识别。在一个给定的应用中，I^2C可以为主模式也可以为从模式。在从模式中，I^2C寻找它自身的从地址和通用调用地址。如果检测到其中一个地址，将产生中断请求。当微控制器希望成为总线主机时，硬件在进入主模式前一直等待，直到总线释放。这样就不会中断一个可能的从机动作。如果在主模式中总线仲裁丢失，I^2C将立即切换到从模式并能在同一个串行传输中检测自身的从地址。



从发送器模式的格式

I2C 主机基本操作方法

(1)设置I2C引脚连接；

(2)设置I2C时钟速率（I2SCLH, I2SCLL）；

(3)设置为主机，并当发送其实信号(I2CONSET的I2EN=1，STA，AA=0)；

(4)发送从机地址，控制I2CONSET发送；

(5)判断总线状态（I2STAT），进行数据传输控制；

(6)发送结束信号(I2CONSET)

I2C从机基本操作方法

(1)设置I2C引脚连接；

(2)设置自身的从几地址（I2ADR）

(3)使能I2C（I2EN=1,AA=1）

(4)判断SI位或者等待I2C中断

(5)判断总线状态（I2STAT），进行数据传输控制；

I2C操作示例

设置I2C总线时钟

```
I2SCLH = (Fpclk/Fi2c + 1) / 2;   //设定I2C时钟
I2SCLL = (Fpclk/Fi2c)/2;
```

初始化I2C为主模式

```
I2C1CONCLR = 0x2C;
I2C1CONSET = 0x60;                    // 设置为主机，并启动总线
```

初始化I2C为从模式

```
I2ADR = adr&0xFE;                     // 设置从机地址
I2CONCLR = 0x28;
I2CONSET = 0x44;                      // I2C配置为从机模式
```

SPI 模块要点解析

SPI 功能概述

SPI0和SPI1是一个全双工的串行接口。它们设计成可以处理在一个给定总线上多个互连的主机和从机。在一定数据传输过程中，接口上只能有一个主机和一个从机能够通信。在一次数据传输中，主机总是向从机发送一个字节数据，而从机也总是向主机发送一个字节数。

SPI 基本特性

两个完全独立的SPI控制器

遵循串行外设接口(SPI)规范

同步、串行、全双工通信

组合的SPI主机和从机

最大数据位速率为输入时钟速率的1/8

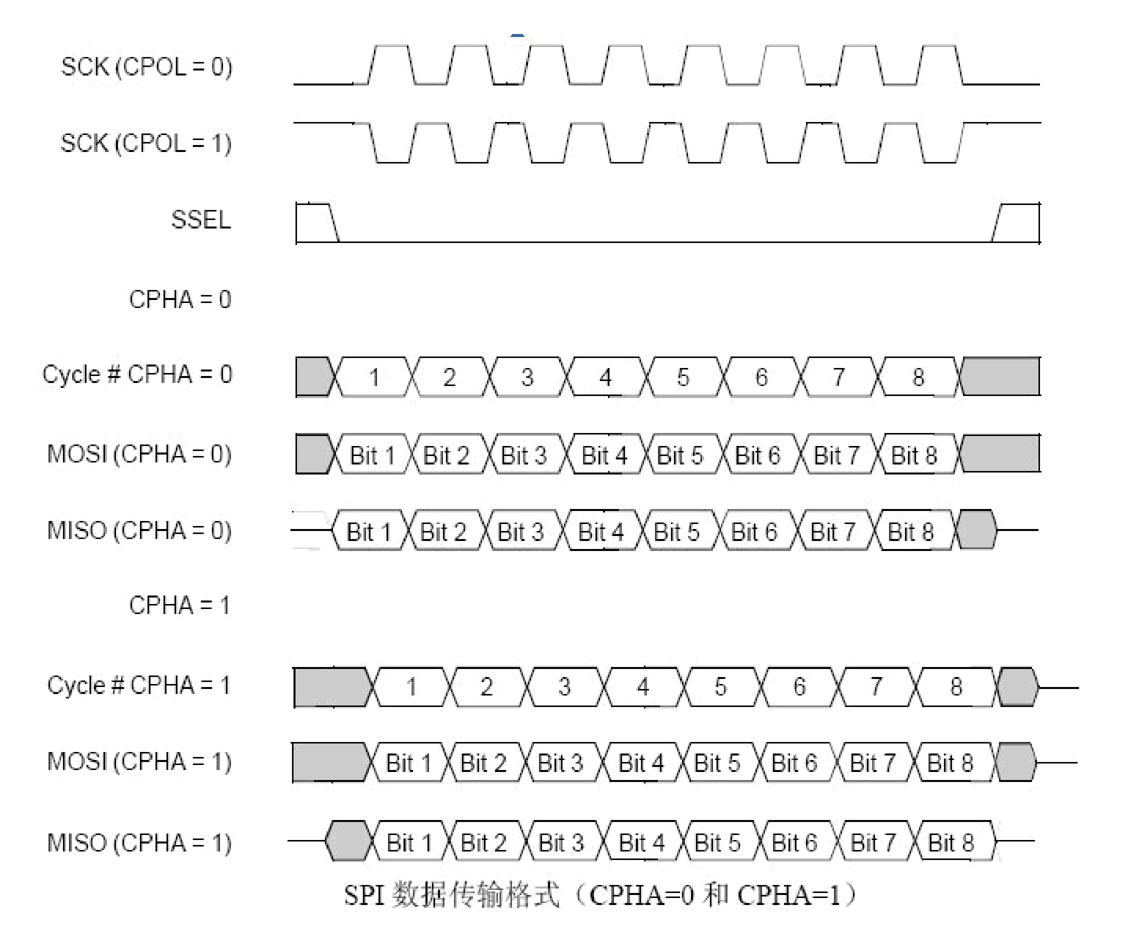
SPI 基本操作方法

SPI 总共有 4 种不同数据传输格式的时序。如下图，时序图描述的是 8 位数据的传输。需要注意的是，该时序图分成了 3 个水平的部分。第一部分描述 SCK 和 SSEL 信号。第二部分描述了 CPHA=0 时的 MOSI 和 MISO 信号。第三部分描述了 CPHA＝1 时的 MOSI 和 MISO 信号。

SPI 数据和时钟的相位关系

CPOL 和 CPHA 的设定	驱动的第一个数据	驱动的其它数据	采样的数据
CPOL=0, CPHA=0	在第一个 SCK 上升沿之前	SCK 下降沿	SCK 上升沿
CPOL=0, CPHA=1	第一个 SCK 上升沿	SCK 上升沿	SCK 下降沿
CPOL=1, CPHA=0	在第一个 SCK 下降沿之前	SCK 上升沿	SCK 下降沿
CPOL=1, CPHA=1	第一个 SCK 下降沿	SCK 下降沿	SCK 上升沿

在时序图的第一部分需要注意两点。第一，时序图包含了CPOL设置为0和1的情况。第二，SSEL信号的激活和未激活。当CPHA=1时，SSEL信号在数据传输之间时总是保持未激活状态。当CPHA=0时则不能保证这一点（信号有可能保持激活状态）。



SPI 数据传输格式（CPHA=0 和 CPHA=1）

数据和时钟的相位关系在表113中描述。该表汇集了下面情况下CPOL和CPHA的每一种设定：

当驱动第一个数据位时

当驱动所有其它数据位时

当采样数据时

8 位传输起始和停止时间的定义取决于器件为主机还是从机，以及 CPHA 变量的设定。

当器件为主机时，传输的起始由包含发送数据字节的主机来指示。此时，主机可激活时钟并开始传输。当传输的最后一个时钟周期结束时，传输结束。

当器件为从机并且CPHA=0时，传输在SSEL信号激活时开始，并在SSEL变为高电平时结束。当器件为从机且CPHA=1时，如果该器件被选择，传输从第一个时钟沿开始，并在数据采样的最后一个时钟沿结束。

SPI 外设描述

有4个寄存器控制SPI外设，SPI控制寄存器包含一些可编程位来控制SPI模块的功能。该寄存器必须在数据传输之前进行设定。

SPI状态寄存器包含只读位，用于监视SPI外设的状态，包括一般性功能和异常状况。该寄存器的主要用途是检测数据传输的完成，这通过SPIF位来实现。其它位用于指示异常状况。异常情况将在后面描述。

SPI数据寄存器用于提供发送和接收的数据字节。串行数据实际的发送和接收通过内部移位寄存器来实现。在发送时向SPI数据寄存器写入数据。数据寄存器和内部移位寄存器之间没有缓冲区。写数据寄存器会使数据直接进入内部移位寄存器。因此数据只能在没有执行数据发送时写入该寄存器。读数据带有缓冲区。当传输结束时，接收到的数据转移到一个单字节的数据缓冲区，下次传输时将其读出。读SPI数据寄存器将返回读数据缓冲区的值。

当SPI模块处于主模式时，SPI时钟计数器寄存器用于控制时钟速率。该寄存器必须在数据传输之前设定。当SPI模块处于从模式时，该寄存器无效。

SPI所使用的I/O口为标准CMOS I/O口。设计上没有实现开漏SPI选项。当器件设置为从机时，它的I/O口只有在被有效的SSEL信号选择时才有效。

SPI 主机操作流程

下面的步骤描述了 SPI 设置为主机时如何处理数据传输。该处理假设任何之前的数据传输已经结束。

1. 将 SPI 时钟计数寄存器设置为所需要的值。
2. 将SPI控制寄存器设置为所需要的设定。
3. 将要发送的数据写入SPI数据寄存器。此写操作启动SPI数据传输。
4. 等待SPI状态寄存器中的SPIF位置位。SPIF位将在SPI数据传输的最后一个周期之后置位。
5. 读取SPI状态寄存器。
6. 从SPI数据寄存器中读出接收到的数据（可选）
7. 如果有更多数据需要发送，则跳到第3步。

注：读或写SPI数据寄存器用来清零SPIF状态位。因此，如果不执行可选的SPI数据寄存器读操作，则需要执行该寄存器的写操作以清零SPIF状态位。

SPI 从机操作流程

下面的步骤描述了SPI设置为从机时如何处理数据传输。该处理假设任何之前的数据传输已经结束。要求驱动SPI逻辑的系统时钟速度至少8倍于SPI。

1. 将SPI控制寄存器设置为所需要的设定。
2. 将要发送的数据写入SPI数据寄存器（可选）。注意这只能在从SPI传输没有进行时执行。
4. 读取SPI状态寄存器。
5. 从SPI数据寄存器中读出接收到的数据（可选）。
6. 如果有更多数据需要发送，则跳到第2步。

备注：读或写SPI数据寄存器用来清零SPIF状态位。因此至少需要执行一个该寄存器的读或写操作来清。

SPI 操作示例

```
/************************************************************************
** 函数名称 ：SPI_Init()
** 函数功能 ：初始化SPI接口，设置为从机。
** 入口参数 ：无
** 出口参数 ：无
************************************************************************/
void  SPI_Init(void)
{
   S0PCR  = (0 << 3)|            // CPHA = 0, 数据在SCK 的第一个时钟沿采样
            (1 << 4) |           // CPOL = 1, SCK 为低有效
            (0 << 5) |           // MSTR = 0, SPI 处于从模式
            (0 << 6) |           // LSBF = 0, SPI 数据传输MSB (位7)在先
            (1 << 7);            // SPIE = 1, SPI 中断被使能
}
/************************************************************************
** 函数名称 ：SSP_Init()
** 函数功能 ：将SSP控制器初始化SPI接口，设置为主机。
** 入口参数 ：无
** 出口参数 ：无
************************************************************************/
void  SSP_Init(void)
{
    SSPCR0 = (0x01 << 8) |              // SCR  设置SPI时钟分频
             (0x00 << 7) |              // CPHA 时钟输出相位,仅SPI模式有效
             (0x01 << 6) |              // CPOL 时钟输出极性,仅SPI模式有效
             (0x00 << 4) |
 // FRF  帧格式 00=SPI,01=SSI,10=Microwire,11=保留
             (0x07 << 0);
 // DSS  数据长度,0000-0010=保留,0011=4位,0111=8位,1111=16位
    SSPCR1 = (0x00 << 3) |              // SOD  从机输出禁能,1=禁止,0=允许
             (0x00 << 2) |              // MS   主从选择,0=主机,1=从机
             (0x01 << 1) |          // SSE  SSP使能,1=允许SSP与其它设备通信
             (0x00 << 0);                 // LBM  回写模式

    SSPCPSR = 0x52;                       // PCLK分频值
 //  SSPIMSC = 0x07;                        // 中断屏蔽寄存器
    SSPICR  = 0x03;                       // 中断清除寄存器
```

定时器使用要点分析

定时器功能概述

LPC2132拥有2个32位可编程定时/计时器，均具有4路捕获和4路比较匹配与输出电路，定时器对外设时钟（pclk）周期进行计数，可选择产生中断或根据4个匹配寄存器的设定，在到达指定的定时值时执行其它动作。它还包括4个捕获输入，用于在输入信号发生跳变时捕获定时器值，并可选择产生中断。

定时器基本特性

带可编程32位预分频器的32位定时器/计数器

具有多达4路32位的捕获通道。当输入信号跳变时可取得定时器的瞬时值。也可选择使捕获事件产生中断。

4个32位匹配寄存器：

—匹配时定时器继续工作，可选择产生中断

—匹配时停止定时器，可选择产生中断

—匹配时复位定时器，可选择产生中断

多达4个对应于匹配寄存器的外部输出，具有下列特性：

—匹配时设置为低电平

—匹配时设置为高电平

—匹配时翻转

—匹配时无动作

定时器主要，用于对内部事件进行计数的间隔定时器，通过捕获输入实现脉宽调制以及自由运行的定时器 。

定时器基本操作方法：

(1)计算定时器的时钟频率，设置PR寄存器进行分频操作；

(2)设置比较匹配通道的初始值和工作方式，

(3)若使用定时器的相关中断，需要设置VIC，并使能中断；

(4)设置TCR，启动定时器

其中定时器时钟频率计算公式如下：

$$计数时钟频率 = \frac{Fpclk}{N+1}$$

其中，N 为 PR 的值。

定时器 0 操作示例

```
IRQEnable();                    /* IRQ中断使能*/
/* 定时器0初始化 */
T0TC    = 0;                    /* 定时器设置为0*/
T0PR    = 0;    /* 时钟不分频：定时器计数器每个PCLK周期加1，T0PR = 1时定时器计数器每2个PCLK周期加1*/
T0MCR   = 0x03;        /* 设置T0MR0匹配后复位T0TC，并产生中断标志*/
T0MR0   = Fpclk /2;    /* 0.5秒钟定时 */
T0TCR   = 0x01;        /* 启动定时器*/
/* 设置定时器0中断IRQ */
VICIntSelect = 0x00;          /* 所有中断通道设置为IRQ中断*/
VICVectCntl0 = 0x20 | 0x04; /* 设置定时器0中断通道分配最高优先级*/
VICVectAddr0 = (uint32)IRQ_Timer0; /* 设置中断服务程序地址*/
VICIntEnable = 1 << 0x04;          /* 使能定时器0中断*/
```

备注：相关寄存器请详细参考LPC2132芯片文档。

PWM 脉宽调制器要点解析

PWM 概述

LPC2132的脉宽调制器建立在标准定时器 0/1 之上。应用在 PWM 和匹配功 能当中进行选择。 PWM基于标准的定时器模块并具有其所有特性，不过LPC21324只将

其PWM功能输出到管脚。定时器对外设时钟(pclk)进行计数，可选择产生中断或基于7个匹配寄存器，在到达指定的定时值时执行其它动作。它还包括4个捕获输入，用于在输入信号发生跳变时捕获定时器值，并可选择在事件发生时产生中断。PWM功能是一个附加特性，建立在匹配寄存器事件基础之上。

独立控制上升和下降沿位置的能力使PWM可以应用于更多的领域。如多相位电机控制通常需要3个非重叠的PWM输出，而这3个输出的脉宽和位置需要独立进行控制。

两个匹配寄存器可用于提供单边沿控制的PWM输出。一个匹配寄存器（PWMMR0）通过匹配时重新设置计数值来控制PWM周期率。另一个匹配寄存器控制PWM边沿的位置。每个额外的单边沿控制PWM输出只需要一个匹配寄存器，因为所有PWM输出的重复率速率是相同的。多个单边沿控制PWM输出在每个PWM周期的开始，当PWMMR0发生匹配时，都有一个上升沿。

3个匹配寄存器可用于提供一个双边沿控制PWM输出。也就是说，PWMMR0匹配寄存器控制PWM周期速率，其它匹配寄存器控制两个PWM边沿位置。每个额外的双边沿控制PWM输出只需要两个匹配寄存器，因为所有PWM输出的重复率速率是相同的。

使用双边沿控制PWM输出时，指定的匹配寄存器控制输出的上升和下降沿。这样就产生了正脉冲（当上升沿先于下降沿时）和负脉冲（当下降沿先于上升沿时）。

与PWM相关的管脚

管脚名称	管脚方向	管脚描述	
PWM1	输出	PWM 通道 1 输出	P0.0
PWM2	输出	PWM 通道 2 输出	P0.7
PWM3	输出	PWM 通道 3 输出	P0.1
PWM4	输出	PWM 通道 4 输出	P0.8
PWM5	输出	PWM 通道 5 输出	P0.21
PWM6	输出	PWM 通道 6 输出	P0.9

PWM 基本特性

7个匹配寄存器，可实现6个单边沿控制或3个双边沿控制PWM输出，或这两种类型的混合输出：

—连续操作，可选择在匹配时产生中断

－匹配时停止定时器，可选择产生中断

－匹配时复位定时器，可选择产生中断

每个匹配寄存器对应一个外部输出，具有下列特性：

－匹配时设置为低电平

－匹配时设置为高电平

－匹配时翻转

－匹配时无动作

支持单边沿控制和/或双边沿控制的PWM输出。单边沿控制PWM输出在每个周期开始时总是为高电平，除非输出保持恒定低电平。双边沿控制PWM输出可在一个周期内的任何位置产生边沿。这样可同时产生正和负脉冲。

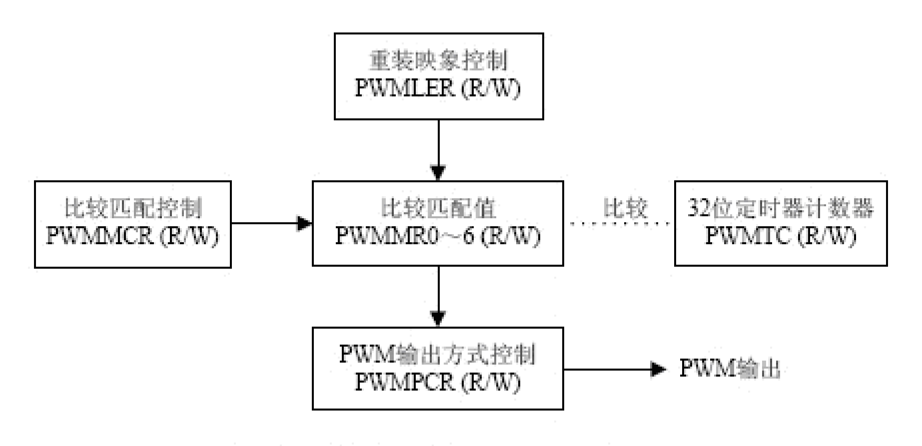
脉冲周期和宽度可以是任何的定时器计数值。这样可实现灵活的分辨率和重复速率的设定。所有PWM输出都以相同的重复率发生。

双边沿控制的PWM输出可编程为正脉冲或负脉冲

匹配寄存器更新与脉冲输出同步，防止产生错误的脉冲。软件必须在新的匹配值生效之前将它们释放。　如果不使能PWM模式，可作为一个标准定时器

带可编程32位预分频器的32位定时器/计数器

当输入信号跳变时4个捕获寄存器可取得定时器的瞬时值，也可选择使捕获事件产生中断。



PWM 的比较匹配寄存器功能框图

如 图 定时器比较匹配由控制寄存器PWMMCR进行匹配操作设置，而PWMMR0～6 则为 7 路比较匹配通道的比较值寄存器。当比较匹配时，将会按照PWMMCR设置的方法产生中断或复位PWMTC等，而且PWMPCR可以控制单边/双边PWM输出，允许/不允许PWM输出。另外，为了确保对PWMMR0～6 的比较值进行修改过程中不影响PWM输出，使用了一个PWMLER锁存使能寄存器，当要修改MR0～6 的比较值时，只有控制PWMLER的对应位置位，在匹配 0 事件发生后此值才会生效。

PWM 基本操作方法：

- 连接 PWM 功能管脚输出，即设置 PINSEL0、PINSEL1；
- 设置 PWM 定时器的时钟分频值 (PWMPR)，得到所要的定时器时钟；
- 设置比较匹配控制(PWMMCR)，并设置相应比较值(PWMMRx)；
- 设置 PWM 输出方式并允许 PWM 输出(PWMPCR)及锁存使能控制(PWMLER)；
- 设置 PWMTCR，启动定时器，使能 PWM；
- 运行过程中要更改比较值时，更改之后要设置锁存使能。

使用双边沿 PWM 输出时，建议使用 PWM2、PWM4、PWM6；使用单边 PWM 输出时，在 PWM 周期开始时为高电平，匹配后为低电平，使用 PWMMR0 作为 PWM 周期控制，PWMMRx 作为占空比控制。

PWM 寄存器映射

名称	描述	访问	复位值	地址
PWMIR	**PWM 中断寄存器** 读取 IR 可识别中断源，对应位写入 1 将清除相应中断。	R/W	0	0xE0014000
PWMTCR	**PWM 定时器控制寄存器** TCR 使能或复位定时器计数器功能。	R/W	0	0xE0014004
PWMTC	**PWM 定时器计数器** 32 位 TC 每经过 PR+1 个 *pclk* 周期加 1。TC 通过 TCR 进行控制。	R/W	0	0xE0014008
PWMPR	**PWM 预分频寄存器** TC 每经过 PR+1 个 *pclk* 周期加 1。	R/W	0	0xE001400C
PWMPC	**PWM 预分频计数器** 每当 32 位 PC 的值增加到等于 PR 中保存的值时，TC 加 1。	R/W	0	0xE0014010
PWMMCR	**PWM 匹配控制寄存器** MCR 用于控制在匹配时是否产生中断或复位 TC。	R/W	0	0xE0014014
PWMMR0	**PWM 匹配寄存器 0** MR0 可通过 MCR 设定为在匹配时复位 TC，停止 TC 和 PC 和/或产生中断。此外，MR0 和 TC 的匹配将置位所有单边沿模式的 PWM 输出，并置位双边沿模式下的 PWM1 输出。	R/W	0	0xE0014018
PWMMR1	**PWM 匹配寄存器 1** MR1 可通过 MCR 设定为在匹配时复位 TC，停止 TC 和 PC 和/或产生中断。此外，MR1 和 TC 的匹配将清零单边沿模式或双边沿模式下的 PWM1，并置位双边沿模式下的 PWM2 输出。	R/W	0	0xE001401C
PWMMR2	**PWM 匹配寄存器 2** MR2 可通过 MCR 设定为在匹配时复位 TC，停止 TC 和 PC 和/或产生中断。此外，MR2 和 TC 的匹配将清零单边沿模式或双边沿模式下的 PWM2，并置位双边沿模式下的 PWM3 输出。	R/W	0	0xE0014020

定时器控制寄存器（PWMTCR – 0xE0014004）

PWMTCR	功能	描述	复位值
0	计数器使能	为 1 时，PWM 定时器计数器和 PWM 预分频计数器使能计数。 为 0 时，计数器被禁止。	0
1	计数器复位	为 1 时，PWM 定时器计数器和 PWM 预分频计数器在 pclk 的下一个上升沿同步复位。计数器在 TCR[1]恢复为 0 之前保持复位状态。	0
2	保留		NA
3	PWM 使能	为 1 时，PWM 模式使能。PWM 模式将映像寄存器连接到匹配寄存器。只有在 PWMLER 中的相应位置位后发生的匹配 0 事件才会使程序写入匹配寄存器的值生效。需要注意的是，决定 PWM 速率（PWM 匹配 0）的匹配寄存器必须在使能 PWM 之前设定。否则不会发生使映像寄存器内容生效的匹配事件。	0

名称	描述	访问	复位值	地址
PWMMR3	**PWM 匹配寄存器 3** MR3 可通过 MCR 设定为在匹配时复位 TC，停止 TC 和 PC 和/或产生中断。此外，MR3 和 TC 的匹配将清零单边沿模式或双边沿模式下的 PWM3，并置位双边沿模式下的 PWM4 输出。	R/W	0	0xE0014024
PWMMR4	**PWM 匹配寄存器 4** MR4 可通过 MCR 设定为在匹配时复位 TC，停止 TC 和 PC 和/或产生中断。此外，MR4 和 TC 的匹配将清零单边沿模式或双边沿模式下的 PWM4，并置位双边沿模式下的 PWM5 输出。	R/W	0	0xE0014040
PWMMR5	**PWM 匹配寄存器 5** MR5 可通过 MCR 设定为在匹配时复位 TC，停止 TC 和 PC 和/或产生中断。此外，MR5 和 TC 的匹配将清零单边沿模式或双边沿模式下的 PWM5，并置位双边沿模式下的 PWM6 输出。	R/W	0	0xE0014044
PWMMR6	**PWM 匹配寄存器 6** MR6 可通过 MCR 设定为在匹配时复位 TC，停止 TC 和 PC 和/或产生中断。此外，MR6 和 TC 的匹配将清零单边沿模式或双边沿模式下的 PWM6。	R/W	0	0xE0014048
PWMPCR	**PWM 控制寄存器** 使能 PWM 输出并选择 PWM 通道类型为单边沿或双边沿控制。	R/W	0	0xE001404C
PWMLER	**PWM 锁存使能寄存器** 使能使用新的 PWM 匹配值。	R/W	0	0xE0014050

PWM锁存使能寄存器—PWM Latch Enable Register（PWMLER – 0xE0014050）

当 PWM 匹配寄存器用于产生 PWM 时，PWM 锁存使能寄存器用于控制 PWM 匹配寄存器的更新。当定时器处于 PWM 模式时如果软件对 PWM 匹配寄存器位置执行写操作，写入的值将保存在一个映像寄存器中。当 PWM 匹配 0 事件发生时（在 PWM 模式下，通常也会复位定时器），如果对应的锁存使能寄存器位已经置位，那么映像寄存器的内容将传送到实际的匹配寄存器中。此时，新的值将生效并决定下一个 PWM 周期。当发生新值传送时，LER 中的所有位都自动清零。在 PWMLER 中相应位置位和 PWM 匹配 0 事件发生之前，任何写入 PWM 匹配寄存器的值都不会影响 PWM 操作。

例如，当 PWM2 配置为双边沿操作并处于运行中时，改变定时的典型事件顺序如下：

- 将新值写入 PWM 匹配 1 寄存器；
- 将新值写入 PWM 匹配 2 寄存器；
- 写 PWMLER，同时置位 bit1 和 bit2；
- 更改的值将在下一次定时器复位时（当 PWM 匹配 0 事件发生时）生效。

写两个 PWM 匹配寄存器的顺序并不重要，因为在写 PWMLER 之前，写入的新匹配值都无效。这样就确保了两个值同时生效。如果使用单个值，也可用同样的方法更改。

PWMLER中所有位的功能如 表 4.71 所示。

PWM 锁存使能寄存器

PWMLER	功能	描述	复位值
0	使能 PWM 匹配 0 锁存	将该位置位允许最后写入 PWM 匹配 0 寄存器的值在由 PWM 匹配事件引起的下次定时器复位时生效。见 PWM 匹配控制寄存器（PWMMCR）的描述。	0
1	使能 PWM 匹配 1 锁存	将该位置位允许最后写入 PWM 匹配 1 寄存器的值在由 PWM 匹配事件引起的下次定时器复位时生效。见 PWM 匹配控制寄存器（PWMMCR）的描述。	0
2	使能 PWM 匹配 2 锁存	将该位置位允许最后写入 PWM 匹配 2 寄存器的值在由 PWM 匹配事件引起的下次定时器复位时生效。见 PWM 匹配控制寄存器（PWMMCR）的描述。	0
3	使能 PWM 匹配 3 锁存	将该位置位允许最后写入 PWM 匹配 3 寄存器的值在由 PWM 匹配事件引起的下次定时器复位时生效。见 PWM 匹配控制寄存器（PWMMCR）的描述。	0
4	使能 PWM 匹配 4 锁存	将该位置位允许最后写入 PWM 匹配 4 寄存器的值在由 PWM 匹配事件引起的下次定时器复位时生效。见 PWM 匹配控制寄存器（PWMMCR）的描述。	0
5	使能 PWM 匹配 5 锁存	将该位置位允许最后写入 PWM 匹配 5 寄存器的值在由 PWM 匹配事件引起的下次定时器复位时生效。见 PWM 匹配控制寄存器（PWMMCR）的描述。	0
6	使能 PWM 匹配 6 锁存	将该位置位允许最后写入 PWM 匹配 6 寄存器的值在由 PWM 匹配事件引起的下次定时器复位时生效。见 PWM 匹配控制寄存器（PWMMCR）的描述。	0
7	保留		NA

公司名称：杭州威步科技有限公司 联系电话：0571-86576692 18958007935
传真：0571-86576692 E-mail: wenming_hu2002@yahoo.com

PWM控制寄存器—PWM Control Register（PWMPCR – 0xE001404C）

PWM控制寄存器用于使能并选择每个PMW通道的类型。

PWM 控制寄存器

PWMPCR	功能	描述	复位值
1:0	保留		NA
2	PWMSEL2	为 0 时，PWM2 选择单边沿控制模式；为 1 时，选择双边沿控制模式。	0
3	PWMSEL3	为 0 时，PWM3 选择单边沿控制模式；为 1 时，选择双边沿控制模式。	0
4	PWMSEL4	为 0 时，PWM4 选择单边沿控制模式；为 1 时，选择双边沿控制模式。	0
5	PWMSEL5	为 0 时，PWM5 选择单边沿控制模式；为 1 时，选择双边沿控制模式。	0
6	PWMSEL6	为 0 时，PWM6 选择单边沿控制模式；为 1 时，选择双边沿控制模式。	0
8:7	保留		NA
9	PWMENA1	为 1 时，使能 PWM1 输出；为 0 时，禁止 PWM1 输出。	0
10	PWMENA2	为 1 时，使能 PWM2 输出；为 0 时，禁止 PWM2 输出。	0
11	PWMENA3	为 1 时，使能 PWM3 输出；为 0 时，禁止 PWM3 输出。	0
12	PWMENA4	为 1 时，使能 PWM4 输出；为 0 时，禁止 PWM4 输出。	0
13	PWMENA5	为 1 时，使能 PWM5 输出；为 0 时，禁止 PWM5 输出。	0
14	PWMENA6	为 1 时，使能 PWM6 输出；为 0 时，禁止 PWM6 输出。	0
15	保留		NA

PWM 操作示例

```
 PINSEL1  = 1 << 10;      // 设置 P0.21 对应为 PWM5 功能
/* PWM 初始化 */
PWMPR    = 0x00;          // PWMPR = 0x00 时，PWMTC 每 1 个 PCLK 周期加//
    不分频，计数频率为 Fpclk，PWMPR = 1 时，PWMTC 每 2 个 PCLK 周期加 1，
PWMMCR   = 0x02;         // 设置 PWMMR0 匹配时复位 PWMTC
PWMPCR   = 0x2000;           // 允许 PWM5 输出，单边 PWM
PWMMR0   = Fpclk / 1000;     // 匹配
PWMMR5   = PWMMR0 / 2;       // 50%占空比
PWMLER   = 0x05;             // PWM0 和 PWM5 匹配锁存
PWMTCR   = 0x02;            // 复位 PWMTC
PWMTCR   = 0x09;            // 启动 PWM 输出
while(1)
{for(i = 0; i < sizeof(HCMM); i++)
     {
       PWMMR0   = Fpclk / HCMM[i]; // 设置输出频率
       PWMLER   = 0x21;         // 更新匹配值后，必须锁存
        Delay(HCMM_L[i]);           // 延时，控制播放速度
     }}
```

AD 采集要点解析

AD 功能概述

A/D转换器的基本时钟由VPB时钟提供。可编程分频器可将时钟调整至逐步逼近转换所需的4.5MHz（最大）。完全满足精度要求的转换需要11个这样的时钟

LPC2132 拥有 1 个 10 位 8 路 AD 转换器，启动 AD 转换方式很灵活，既可以单路采集，也可以审核制 BURST 方式对多路信号逐次循环采样。

AD 模块主要用于传感器数据采集。

AD 基本特性

10 位逐次逼近式模数转换器

一个或多个输入的 Burst 转换模式

掉电模式

测量范围：0～3V

10位转换时间>=2.44us

可选择由输入跳变或定时器匹配信号触发转换

AD 模块的寄存器功能

必须在置位 PCONP 中的 PCAD0 位前置位。

从对 CLKDIV 的描述，不难得到 ADC 转换时钟 F_{ADC} 计算公式：

$$F_{ADC} = \frac{Fpclk}{CLKDIV + 1}$$

因而，ADC 时钟分频数 CLKDIV 计算如下：

$$CLKDIV = \frac{Fpclk}{F_{ADC}} - 1$$

A/D 寄存器

名称	描述	访问	复位值	AD0 地址&名称
ADCR	A/D 控制寄存器　A/D 转换开始前，必须写入 ADCR 寄存器来选择工作模式。	读/写	0x0000 0001	0xE003 4000 AD0CR
ADDR	A/D 数据寄存器　该寄存器包含 ADC 的 DONE 位（当 DONE 位为 1 时）和 10 位的转换结果。	读/写	NA	0xE003 4004 AD0DR

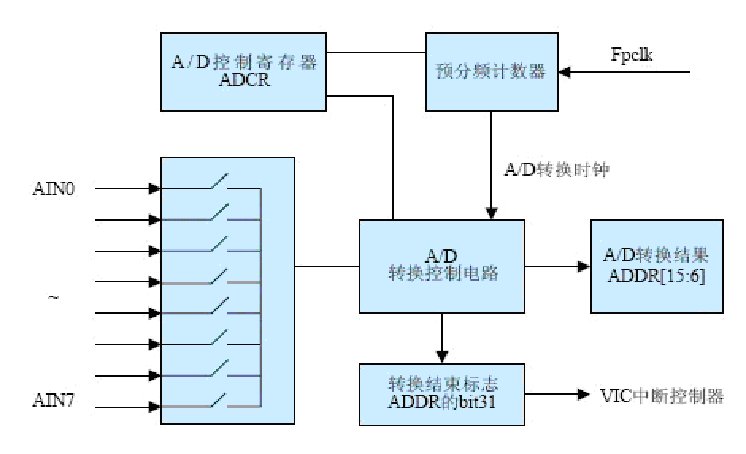


图 4.112　A/D 模块的寄存器功能框图

1. A/D控制寄存器－A/D Control Register（AD0CR – 0xE0034000）
   A/D控制寄存器的控制AD转换通道选择、转换速率、转换精度、起始条件等信息，

A/D数据寄存器－A/D Data Register（AD0DR – 0xE0034004）

AD数据寄存器包含AD转换完成标志和得到的数据等信息

A/D 数据寄存器

ADDR	名称	描述	复位值
5:0	保留		0
15:6	V/V_{3A}	当 DONE 为 1 时，该字段包含一个二进制制数，用来代表 SEL 字段选中的 Ain 脚的电压。该字段根据 VddA 脚上的电压对 Ain 脚的电压进行划分。该字段为 0 表明 Ain 脚的电压小于，等于或接近于 VssA；该字段为 0x3FF 表明 Ain 脚的电压接近于，等于或大于 V_{3A}。为了测试的需要，写入到该字段的数据捕获到移位寄存器，寄存器的移位时钟为 A/D 转换器时钟。仅当 TEST1:0 为 10 时，寄存器的 MS 位供给 A/D 转换器的 DINSERI 输入。	X
23:16	保留	保留，这些位读出时为 0，用户不应将其置位。	0
26:24	CHN	这些位包含的是 LS 位的转换通道。	X
29:27	保留	保留，这些位读出为 0，用户不应将其置位。	0
30	OVERUN	Burst 模式下，如果在转换产生 LS 位的结果前一个或多个转换结果被丢失和覆盖，该位置位。在非 FIFO 操作中，该位通过读 ADDR 寄存器清零。	0
31	DONE	A/D 转换结束时该位置位。该位在 ADDR 被读出和 ADCR 被写入时清零。如果 ADCR 在转换过程中被写入，该位置位，启动一次新的转换。	0

A/D控制寄存器—A/D Control Register（AD0CR – 0xE0034000）

A/D控制寄存器的控制AD转换通道选择、转换速率、转换精度、起始条件等信息

A/D 控制寄存器

ADCR	名称	描述	复位值
7:0	SEL	**输入通道选择** 从 AD0.7:0 管脚中选择采样和转换输入脚。位 0 选择管脚 AD0.0，位 7 选择管脚 AD0.7。软件控制模式下，这些位中只有一位可被置位。硬件扫描模式下，SEL 可为 1～8 中的任何一个值。SEL 为零时等效于为 0x01。	0x01
15:8	CLKDIV	**时钟分频** 将 VPB 时钟（PCK）进行（CLKDIV 的值+1）分频得到 A/D 转换时钟，转换时钟必须小于或等于 4.5MHz。典型地，软件将 CLKDIV 编程为最小值来得到 4.5MHz 或稍低于 4.5MHz 的时钟，但某些情况下（例如高阻抗模拟电源）可能需要更低的时钟。	0
16	BURST	**突发模式** 如果该位为 0，转换由软件控制，需要 11 个时钟方能完成。如果该位为 1，A/D 转换器以 CLKS 字段选择的速率重复执行转换，（如果必要）并从 SEL 字段中为 1 的位对应的管脚开始扫描。A/D 转换器启动后的第一次转换的是 SEL 字段中为 1 的位中的最低有效位对应的模拟输入，然后是为 1 的更高有效位对应的模拟输入（如果可用）。重复转换通过清零该位终止，但该位被清零时并不会中止正在进行的转换。	0
19:17	CLKS	**突发模式时钟选择** 该字段用来选择 Burst 模式下每次转换使用的时钟数和所得 ADDR 转换结果的 LS 位中可确保精度的位的数目，CLKS 可在 11 个时钟（10 位）～4 个时钟（3 位）之间选择： 000：11 个时钟，可确保精度为 10 位 001：10 个时钟，可确保精度为 9 位 … 111：4 个时钟，可确保精度为 3 位	000
21	PDN	**掉电** 1：A/D 转换器处于正常工作模式。 0：A/D 转换器处于掉电模式。	0
23:22	TEST1:0	**器件测试** 这些位用于器件测试。00＝正常模式，01＝数字测试模式，10＝DAC 测试模式，11＝一次转换测试模式。	0
26:24	START	**启动控制** 当 BURST 为 0 时，这些位控制着 A/D 转换是否启动和何时启动： 000：不启动（PDN 清零时使用该值） 001：立即启动转换 010：EDGE 选择的边沿出现在 P0.16 脚时启动转换 011：EDGE 选择的边沿出现在 P0.22 脚时启动转换 注意：START 选择 100-111 时 MAT 信号不必输出到管脚上 100：EDGE 选择的边沿出现在 MAT0.1 时启动转换 101：EDGE 选择的边沿出现在 MAT0.3 时启动转换 110：EDGE 选择的边沿出现在 MAT1.0 时启动转换 111：EDGE 选择的边沿出现在 MAT1.1 时启动转换	000
27	EDGE	**边沿选择** 该位只有在 START 字段为 010～111 时有效。 0：在所选 CAP/MAT 信号的下降沿启动转换 1：在所选 CAP/MAT 信号的上升沿启动转换	0
31:28	保留		0

注：关于掉电，进入掉电时，PDN 必须在清零 PCONP 的 PCAD0 位前清零，而退出掉电时，PDN 位

AD 基本操作方法

(1)硬件触发转换:如果 ADCR 的 BURST 位为 0 且 START 字段的值包含在 010-111 之内，当所选管脚或定时器匹配信号发生跳变时 A/D 转换器启动一次转换。也可选择在 4 个匹配信号中任何一个的指定边沿转换,或者在 2 个捕获/匹配管脚中任何一个的指定边沿转换。将所选端口的管脚状态或所选的匹配信号与 ADCR 位 27 相异或用作边沿检测逻辑。

(2)时钟产生:一般，我们很希望时钟分频器（利用它来得到 4.5MHz 的转换时钟）在 A/D 转换器空闲时保持复位状态，以便在 ADCR 的 START 字段被写入 01 或所选边沿出现在选择的信号上时可立刻启动采样时钟。这个特性可以节省功率，尤其适用于 A/D 转换器很少使用的场合。 中断 当 DONE 位为 1 时，中断请求声明到向量中断控制器（VIC）。软件通过 VIC 中 A/D 转换器的中断使能位来控制是否产生中断。DONE 在 ADDR 读出时清零。

(3)精度和数字接收器:当A/D转换器用来测量Ain脚的电压时,并不理会管脚在管脚选择寄存器中的设置（见管脚连接模块），但是通过禁能管脚的数字接收器来选择Ain功能可以提高转换精度。

AD 操作示例

```
/*************************************************************************
**  AD 相关 IO 端口设置
*************************************************************************/

void AD_IO_SET()
{
   PINSEL1 = (PINSEL1&0xCFFFFFFF) | 0x10000000 ; // 设置P0.30对应AD功能
}

/*************************************************************************
** 函数名称 : AD_INT()
** 函数功能 : AD 采集初始化设置
*************************************************************************/
```

```
void AD_int()
{
    /* 进行 ADC 模块设置，AD0CR 为 AD 控制寄存器 */
    AD_IO_SET();
    AD0CR = (1 << 3)                        |   // SEL=8,选择通道 3(AD0.3)
        ((Fpclk / 1000000 - 1) << 8)        |
// CLKDIV=Fpclk/1000000-1,转换时钟为 1MHz
           (0 << 16)                        |   //BURST=0,软件控制转换操作，需
要 11 个时钟方能完成
           (0 << 17)                        |   // CLKS=0, 使用 11clock 转换
           (1 << 21)                        |   // PDN=1, 正 常 工 作 模 式 ， 若
PDN=0,A/D 转换处于掉电模式
           (0 << 22)                        |   // TEST1:0=00,正常工作模式
           (1 << 24)                        |   // START=1,直接启动 ADC 转换
           (0 << 27);                           // 直接启动 ADC 转换时,此位无效
    DelayNS(10);
    ADC_Data = AD0DR;     // 读取 ADC 结果，并清除 DONE 标志位
}
/***********************************************************************
** 函数名称 ：AD_GET()
** 函数功能 ：AD 转换
***********************************************************************/
uint32 AD_GET()
{
    uint32  ADC_OUT;
    AD0CR |= 1 << 24;                        // 进行第一次转换
    while ((ADDR & 0x80000000) == 0);          // 等待转换结束
    AD0CR |= 1 << 24;                        // 再次启动转换
    while ((AD0DR & 0x80000000) == 0);         // 等待转换结束
    ADC_Data = AD0DR;                        // 读取 ADC 结果
    ADC_Data = (ADC_Data >> 6) & 0x3ff;
    // 右移 6 位是因为 AD0DR 低 6 位没有用到（3FF：AD0DR 的 15：6）
    ADC_Data = ADC_Data * 2640;
    // 参考电压经过 20/（5+20）=4/5 分压，3300*4/5=2640
    ADC_OUT = ADC_Data / 1024;
    // AD 转换精度为 10 位，也就是 2 的 10 次方=1024
       return(ADC_OUT);
}
```

RTC 时钟要点解析

RTC 功能概述

实时时钟(RTC)提供一套计数器在系统上电和关闭操作时对时间进行测量。RTC 消耗的功率非常低，这使其适合于由电池供电的，CPU不连续工作(空闲模式)的系统

RTC 基本特性

测量保持日历或时钟的时间通路

超低功耗设计，支持电池供电系统

提供秒、分、小时、日、月、年和星期

可编程基准时钟分频器允许调节RTC以适应不同的晶振频率

混合寄存器

地址	名称	规格	描述	访问
0xE0024000	ILR	2	**中断位置寄存器** 读出的该位置寄存器的值指示了中断源，向寄存器的一个位写入 1 来清除相应的中断	R/W
0xE0024004	CTC	15	**时钟节拍计数器** 该寄存器的值来自时钟分频器	RO
0xE0024008	CCR	4	**时钟控制寄存器** 控制时钟分频器的功能	R/W
0xE002400C	CIIR	8	**计数器递增中断寄存器** 当计数器递增时，选择一个计数器产生中断	R/W
0xE0024010	AMR	8	**报警屏蔽寄存器** 控制报警寄存器的屏蔽	R/W

计数器增量中断寄存器位

CIIR	功能	描述
0	IMSEC	为 1 时，秒值的增加产生一次中断
1	IMMIN	为 1 时，分值的增加产生一次中断
2	IMHOUR	为 1 时，小时值的增加产生一次中断
3	IMDOM	为 1 时，日期（月）值的增加产生一次中断
4	IMDOW	为 1 时，星期值的增加产生一次中断
5	IMDOY	为 1 时，日期（年）值的增加产生一次中断
6	IMMON	为 1 时，月值的增加产生一次中断
7	IMYEAR	为 1 时，年值的增加产生一次中断

组别	名称	规格	描述	访问	复位值	地址
混合寄存器组	ILR	2	中断位置寄存器	R/W	*	0xE0024000
	CTC	15	时钟节拍计数器	RO	*	0xE0024004
	CCR	4	时钟控制寄存器	R/W	*	0xE0024008
	CIIR	8	计数器递增中断寄存器	R/W	*	0xE002400C
	AMR	8	报警屏蔽寄存器	R/W	*	0xE0024010
时间寄存器组	CTIME0	(32)	完整时间寄存器 0	RO	*	0xE0024014
	CTIME1	(32)	完整时间寄存器 1	RO	*	0xE0024018
	CTIME2	(32)	完整时间寄存器 2	RO	*	0xE002401C
时间计数器	SEC	6	秒寄存器	R/W	*	0xE0024020
	MIN	6	分寄存器	R/W	*	0xE0024024
	HOUR	5	小时寄存器	R/W	*	0xE0024028
	DOM	5	日期（月）寄存器	R/W	*	0xE002402C
	DOW	3	星期寄存器	R/W	*	0xE0024030
	DOY	9	日期（年）寄存器	R/W	*	0xE0024034
	MONTH	4	月寄存器	R/W	*	0xE0024038
	YEAR	12	年寄存器	R/W	*	0xE002403C
报警寄存器	ALSEC	6	秒报警值	R/W	*	0xE0024060
	ALMIN	6	分报警值	R/W	*	0xE0024064
	ALHOUR	5	小时报警值	R/W	*	0xE0024068
	ALDOM	5	日期（月）报警值	R/W	*	0xE002406C
	ALDOW	3	星期报警值	R/W	*	0xE0024070
	ALDOY	9	日期（年）报警值	R/W	*	0xE0024074
	ALMON	4	月报警值	R/W	*	0xE0024078
	ALYEAR	12	年报警值	R/W	*	0xE002407C
预分频器	PREINT	13	预分频值，整数部分	R/W	0	0xE0024080
	PREFRAC	15	预分频值，小数部分	R/W	0	0xE0024084

* RTC 当中除预分频器部分之外的其它寄存器都不受器件复位的影响。如果 RTC 使能，这些寄存器必须通过软件来初始化。

RTC 基本操作方法

(1)选择时钟源（相关寄存器CCR）

(2)设置RTC预分频器（PREINT，PREFRAC）

(3)初始化RTC时钟值，如YEAR,MONTH等

(4)报警中断设置，如CIIR和AMR等

(5)启动RTC（即CCR的CLKEN=1）

(6)读取完整时间寄存器值，或等待中断

RTC 操作示例

```
/*********************************************************************
** 函数名称 ：RTCInit()
** 函数功能 ：初始化实时时钟
*********************************************************************/
void RTCInit (void)
{
   PREINT = Fpclk / 32768 - 1; // 设置基准时钟分频器
   PREFRAC = Fpclk - (Fpclk / 32768) * 32768;
   CCR    = 0x00;             // 禁止时间计数器
   YEAR   = 2010;
   MONTH = 06;
   DOM    = 7;
   DOW    = 1;
   HOUR   = 8;
   MIN    = 30;
   SEC    = 59;
   CIIR = 0x01;           // 设置秒值的增量产生1次中断
   CCR   = 0x01;           // 启动RTC
}
/********************************************************************
** 函数名称 ：SendTimeRtc()
** 函数功能 ：读取RTC的时间值，并将读出的时分秒值通过串口送到上位机显示。
********************************************************************/
void SendTimeRtc (void)
{
   uint32 datas;
   uint32 times;
   uint32 bak;
   times = CTIME0;                          // 读取完整的时钟寄存器
   datas = CTIME1;
   bak = (datas >> 16) & 0xfff;             // 获取 年
   year_data[3]=bak / 1000;
   year_data[2]=bak % 1000/100;
   year_data[1]=bak % 100/10;
   year_data[0]=bak % 10;
   bak = (datas >> 8) & 0x0f;               // 获取 月
   month_data[1]=bak / 10;
   month_data[0]=bak % 10;
   bak = datas & 0x1f;                     // 获取 日
```

```
    day_data[1]=bak / 10;
    day_data[0]=bak % 10;
    week_data[0]= (times >> 24) & 0x07;        // 获取 星期
    bak = (times >> 16) & 0x1f;                // 获取 小时
    hour_data[1]=bak / 10;
    hour_data[0]=bak % 10;
    bak = (times >> 8) & 0x3f;                 // 获取 分钟
    minute_data[1]=bak / 10;
    minute_data[0]=bak % 10;
    bak = times & 0x3f;                    // 获取 秒钟
    second_data[1]=bak / 10;
    second_data[0]=bak % 10;
}
```

RTC 使用注意事项

由于RTC的时钟源为VPB时钟（pclk），时钟出现的任何中断都会导致时间值的偏移。如果RTC初始化成这个时间值或从RTC激活后运行的一段时间内出现了一个错误，它们带来的变化都将影响真实的时钟时间。

在断电时不能保持RTC的状态。如果时钟源丢失、中断或改变，RTC也无法维持时间计数。芯片的断电将使RTC寄存器的内容完全丢失。进入掉电模式会使时间的更新出现误差。在系统操作过程中（重新配置PLL、VPB定时器或RTC预分频器）改变RTC的时间基准会使累加时间出现错误。

中断的产生由中断位置寄存器(ILR)、计数器递增中断寄存器(CIIR)、报警寄存器和报警屏蔽寄存器(AMR) 控制。只有转换到中断状态才能产生中断。ILR单独使能CIIR和AMR中断。CIIR中的每个位都对应一个时间计数器。如果CIIR使能用于一个特定的计数器，那么该计数器的值每增加一次就产生一个中断。报警寄存器允许用户设定产生中断的日期或时间。AMR提供一个屏蔽报警比较的机制。如果所有非屏蔽报警寄存器与它们对应的时间计数器的值相匹配时，则会产生中断。

第三部分 基于LPC2132 嵌入式无线应用

时代需要速度更快、互操作更方便以及更安全可靠的无线网络，Nordic VLSI ASA、Freascale、Atmel等具有国际影响力的IC生厂商都相继推出了新一代短距离无线数据通信收发芯片，以nRF905、CC1100 为主流的无线芯片性能得到了很大提高，最新的无线收发芯片将全部无线通信需要的调制/解调芯片、高/低频放大器等全部集成在芯片中，使外围器件大幅度减少，很容易与各种型号微控制器连接实现高可靠性无线通信，使开发无线产品成本大大降低，开发难度更简单，应用更广泛，嵌入式无线通信和无线网络将逐步取代现有的有线通信和有线网络，无线技术将展示其巨大的影响力，必将掀起一场的新的技术浪潮。

主流无线芯片汇总及特点解析

系列 A: 433/868/915MHZ 频段

1. NRF905 基本特性

工作电压：1.9-3.6V

调制方式： GFSK

接收灵敏度：-100dBm

最大发射功率: 10mW (+10dBm)

最大传输数率：50kbps

瞬间最大工作电流: <30mA

工作频率：(422.4-473.5MHZ)

1)接收发送功能合一，收发完成中断标志

2)433/868/915 工作频段，433MHZ 开放 ISM 频段免许可使用

3)发射速率 50Kbps，选用外置 433 天线，空旷通讯距离可达 300 米左右，加功放可到 3000 米左右；室内通信仍有良好通信效果，3-6 层可实现可靠通信，抗干扰性能强，很强的扰障碍穿透性能；

4)每次最多可发送接收 32 字节,并可软件设置发送/接收缓冲区大小 1/2/4/8/16/32

5)170 个频道，可满足多点通讯和跳频通讯需求，实现组网通讯，TDMA，CDMA，FDMA；

6)内置硬件 8/16 位 CRC 校验，开发更简单，数据传输可靠稳定。

7)1.9-3.6V 工作，低功耗，待机模式仅 2.5uA.

8)内置 SPI 接口，也可通过 I/O 口模拟 SPI 实现。最高 SPI 时钟可达 10M。

2.SI4432 基本特性

1)完整的 FSK 收发器，

2)工作频率范围 430.24~439.75MHz；发射功率最大 17dBm，接收灵敏度-115 dBm（波特率 9.6Kbps）；空旷通讯距离 800 米左右（波特率 9.6Kbps）

3)工作频率范围 900.72~929.27MHz；发射功率最大 17dBm；接收灵敏度-115 dBm（波特率 9.6Kbp）；空旷通讯距离 800 米左右（波特率 9.6Kbps）

4)传输速率最大 128Kbps

5)FSK 频偏可编程(15~240KHz)

6)接收带宽可编程(67~400KHz)

7)SPI 兼容的控制接口，低功耗任务周期模式，自带唤醒定时器

8)+20dB,低的接收电流(18.5mA)，最大发射功率的电流(73mA)

3.CC1100 芯片特性

工作电压：1.8-3.6V

接收灵敏度：在 1200 波特率下-110dBm

最大发射功率: 10mW (+10dBm)

最大传输数率：500kbps

瞬间最大工作电流: <30mA

工作频率：(387-464MHZ)

1）315、433、868、915Mh 的 ISM 和 SRD 频段

2）最高工作速率 500kbps，支持 2-FSK、GFSK 和 MSK 调制方式

选用外置 433 天线，直线通讯距离可达 300 米左右，降低通信波特率距离更远，我公司也提供高精度参数 RF1100SE 模块，性能更佳，室内通信仍有良好通信效果，3-6 层可实现可靠通信，抗干扰性能强，很强的扰障碍穿透性能；

3）高灵敏度（1.2kbps 下-110dDm，1%数据包误码率）

4）内置硬件 CRC 检错和点对多点通信地址控制

5）较低的电流消耗（RX 中，15.6mA，2.4kbps,433MHz）

6）可编程控制的输出功率，对所有的支持频率可达+10dBm

7）支持低功率电磁波激活功能，支持载波侦听系统

8）模块可软件设地址，软件编程非常方便

9）单独的 64 字节 RX 和 TX 数据 FIFO

4.CC1020 芯片特性

1)频率范围为 402 MHz -470MHz 工作

2)高灵敏度（对 12.5kHz 信道可达-118dBm）

3)可编程输出功率，最大 10dBm

4)低电流消耗（RX:19.9mA）

5)低压供电（2.3V 到 3.6V）

6)数据率最高可以达到 153.6Kbaud

7)SPI 接口配置内部寄存器

8)比相同功率下，NRF905- CC1100 远 1/3

5.A7102 基本特性

1)433Mhz 开放 ISM 频段免许可证使用

2)最高工作速率 50kbps，高效 GFSK 调制，抗干扰能力强，适合工业控制场合

3)125 频道，满足多点通信和跳频通信需要

4)内置硬件 CRC 检错和点对多点通信地址控制

5)低功耗 3-3.6V 工作，待机模式下状态仅为 2.5uA

6)收发模式切换时间 < 650us

7)模块可软件设地址，只有收到本机地址时才会输出数据（提供中断指示），可直接与各种单片机使用，软件编程非常方便

8）TX Mode: 在+10dBm 情况下，电流为 40mA; RX Mode: 14mA

9）增加了电源切断模式，可以实现硬件冷启动功能!

10）SPI 接口、功能强大、编程简单，与 RF905SE 编程接口类似。

11）增加了 RSSI 功能，通过 SPI 接口可以获取当前接收到的信号强度(0-255)

系列 B: 2.4GHZ 频段类

1.NRF24L01 芯片特性

工作电压：1.9-3.6V

调制方式： GFSK

最大发射功率: 1mW (0dBm)

最大传输数率：2Mkbps

瞬间最大工作电流: <15mA

工作频率：(2.400-2.524GHZ)

1）接收发送功能合一，收发完成中断标志

2）工作频率 2.4~2.524GHZ，2.4GHZ 开放 ISM 频段免许可使用

3）发射速率最高可达 2Mbps，内置 PCB 天线，空旷通讯距离可达 100 米左右，室内在 50 米左右；抗干扰性能强。

4)每次最多可发送接收 32 字节,并可软件设置发送/接收缓冲区大小 1/2/4/8/16/32

5）125 个频道，可满足多点通讯和跳频通讯需求，可组网通讯，TDMA，CDMA，FDMA；

6）内置硬件 8/16 位 CRC 校验，开发更简单，数据传输可靠稳定。

7）2Mbit/s 速率下@0dBm 输出时的峰值电流 11mA，掉电模式下的功耗 400nA，待机模式下的功耗 32uA，130us 的快速切换和唤醒时间，世界领先的低功耗 nRF24L01 特别适合采用钮扣电池供电的 2.4G 应用,和蓝牙技术相比在提供更高速率的同时只需花更小的功耗。

8）具有片内稳压器，内置 SPI 接口，也可通过 I/O 口模拟 SPI 实现。最高 SPI 时钟可达 10M。

2.NRF2401A 基本特性

工作电压：1.9-3.6V

调制方式： GFSK

最大发射功率: 1mW (0dBm)

最大传输数率：1Mkbps

瞬间最大工作电流: <15mA

工作频率：(2.400-2.524GHZ)

1）2.4Ghz 全球开放 ISM 频段免许可证使用;

2）最高工作速率 1Mbps，高效 GFSK 调制，抗干扰能力强，适合工业控制场合;

3）125 频道，满足多点网络通信需要;

4)内置硬件 8/16 位 CRC 校验和点对多点通信地址控制,结合 TDMA-CDMA-FDMA 原理,可实现无线网络通讯。;

5）低功耗 1.9 - 3.6V 工作，待机模式下状态仅为 1uA ；

模块可软件设地址，只有收到本机地址时才会输出数据（提供中断指示），可和各种单片机使用，软件编程非常方便；

6）收发完成状态提示，每次最多可发 28 字节；

内置专门稳压电路，使用各种电源包括 DC/DC 开关电源均有很好的通信效果；

7）双通道数据接收，内置环行天线，开阔无干扰条件通信距离在 100 米左右。

3.CC2500 基本特性

工作电压：1.8-3.6V

最大发射功率: 1mW (10dBm)

最大传输数率：500kbps

瞬间最大工作电流: <20mA

工作频率：（2.4GHZ-2.484GHZ）

1）2.4GHZ 免费的 ISM 和 SRD 频段

2）最高工作速率 500kbps，支持 2-FSK、GFSK 和 MSK 调制方式

3）高灵敏度

4）内置硬件 CRC 检错和点对多点通信地址控制

5）较低的电流消耗

6）可编程控制的输出功率，对所有的支持频率可达+10dBm

7）支持低功率电磁波激活功能，支持载波侦听系统

8）模块可软件设地址，软件编程非常方便

9）单独的 64 字节 RX 和 TX 数据 FIFO

系列 C： ZIGBEE 协议类

1.CC2420/CC2520 基本特性

(1) 工作在 2400-2483.5 MHz 的 ISM 和 SRD 频段.

-采用直接序列扩频方式.

-工作速率 250kbps，码片速率 2 MChip/s.

-使用 0-QPSK 调制方式.

-高灵敏度（-95dBm）.

-较低的电流消耗（RX :13.3 mA TX:17.4 mA).

-相邻频道干扰能力强(39dB)

-内部集成有 VCO、LNA、PA 以及电源整流器.

-采用低电压供电(2.1～3.6V).

-输出功率编程可控.

(2) IEEE802.15.4-2003 标准 MAC 层硬件支持.

-前导码与同步字段自动生成与检测.

-CRC-16 自动生成与检测.

-空闲信道检测.

-能量检测、接收信号强度与链路质量指示.

-MAC 层具有安全保护(CTR,CBC－MAC,CCM)支持.

(3) 采用 4 线 SPI 标准接口,便于 MCU 配置.

(4) 独立的 128 字节 RX 和 128 字节 TX 数据 FIFO.

公司名称：杭州威步科技有限公司 联系电话：0571-86576692 18958007935
传真：0571-86576692 E-mail: wenming_hu2002@yahoo.com

基于 LPC2132 无线应用案例解析

案例一：无线双向遥控

功能概述

传统智能控制中一般离不开人机交互操作，随着时代发展，越来越多场合需要应用到以无线方式来完成远程控制，而射频技术的飞速发展也使无线应用的可靠性不再是问题，尤其收发功能合一的无线芯片的出现，让无线传输方式可以实现带反馈式传输，而且无线芯片内部集成 CRC 校验功能，使现代无线通信稳定性和抗干扰性能有了质的飞跃。

实验准备

(1)硬件：JASk2132 开发板 1 套

(2 软件：Keil，ADS1.2，串口助手软件

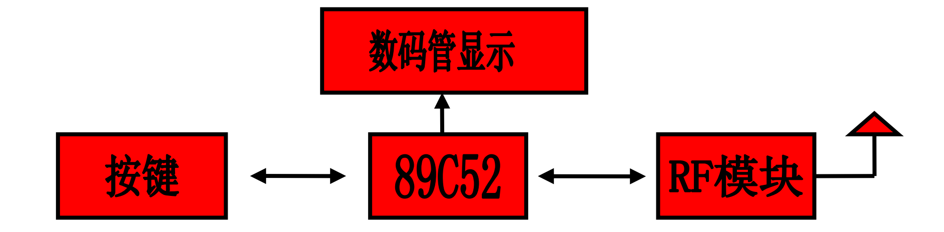
(3)开发工具：J-LINK,USB-TTL 模块，串口线

实验目的：

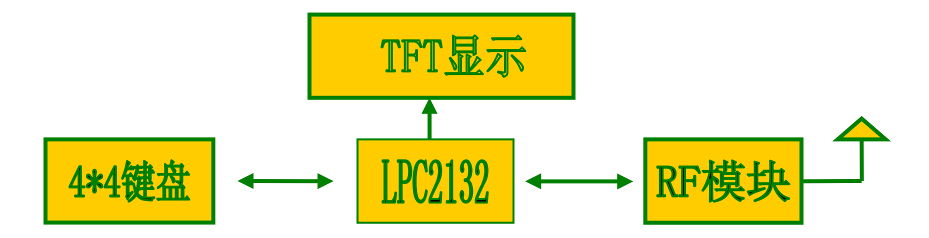
实现两个设备之间相互无线控制，可广泛应用工业遥控等场合

基本工作流程

A: 主/从



B: 主/从



操作步骤：

1. 程序烧写

将该案例开发包配套的 51-JASK2000 代码下载到 JASK2000；

将该案例开发包配套的 ARM-JASK2132 代码下载 JASK2132 开发板中；

2. 硬件连接

A:无线模块与 JASK2000 中 RF 接口对接后上电

B:无线模块与 JASK2132 中 RF 接口对接后上电

3. 检测

当按 JASK2000 底板上的 KEY0，数码管会显示 1，按 KEY1 会显示 2，当 JASK2000 开发板检测到有 KEY0 或 KEY1 时，发送数据，而 JASK2132 测试接收数据，当正确收到数据后对蜂鸣器进行开关操作，TFT 显示不同图片画面。

同理，按 JASK2132 地板上的 S0，S1 按键，JASK2000 开发板收到信息，并显示。

详细请参考 JASK2132 配套光盘实验程序和说明

案例二：无线 232/485 通信设计

功能概述

RS-232 接口设备随处可见，适用于工业数据传输替代传统有线传输，而 RS-232 标准协议也有潜在的不足，主要是距离不能太远，而且在诸多场合布线不方便，所以，无线 232 概念油然而生。

实际应用中，典型的应用：传感器数据采集－单片机处理－无线数据传送－无线接收－单片机读取接收数据－232 串口上传 PC 数据显示及相关数据库。

添加无元素，某种程度上增加 RS-232 协议通信距离，使用高功率无线数传模块，通讯距离可以高达 1KM 以上，可以替代传统有线 232、485 总线，并且免去布线的麻烦，也降低了总体成本，而且维护简单，费用也更低。诚然，232、485 以其简单、稳定、可靠等优势仍发挥着巨大作用，无线通信技术也不能完全取代有线通信的价值，无线更多是起到弥补和完善的作用。

实验准备：

(1)硬件：JASk2132 开发板 1 套

(2 软件：Keil，ADS1.2，串口助手软件

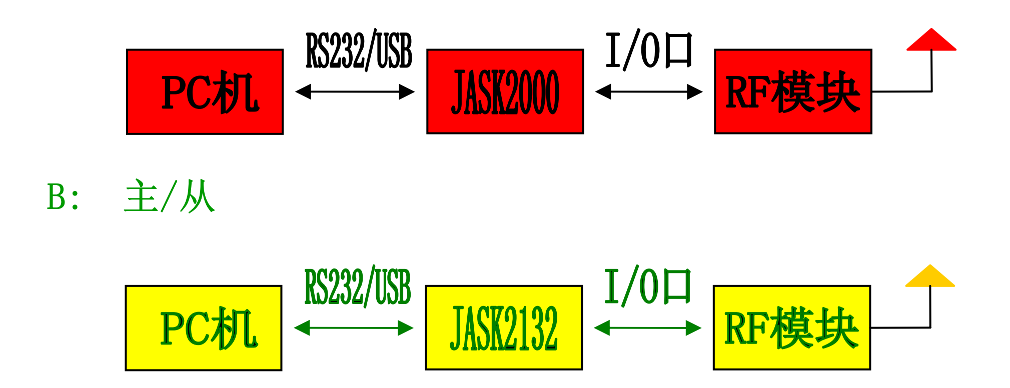
(4)开发工具：J-LINK,USB-TTL 模块，串口线

实验目的：

实现两个电脑之间双向通信，可以应用于 2 个 232 接口设备之间双向通信,任何一方都可以成为主机。

基本该工作流程

A: 主/从



操作步骤：

1.程序烧写

将该案例开发包配套的 51-JASK2000 代码下载到 JASK2000；

将该案例开发包配套的 ARM-JASK2132 代码下载 JASK2132 开发板中；

2.硬件连接

A:按照 PC--USB/TTL 模块--JASK2000 方式连接

B:按照 PC--USB/TTL 模块--JASK2132 方式连接

3.检测

按照打开串口助手软件，按照截图进行操作，可以看到电脑之间无线通信

详细请参考 JASK2132 配套光盘实验程序和说明

案例三：无线温度数据采集

功能概述

温度是诸多领域敏感的参数，温度监控已成了典型应用，本设计具有完全可以应用与实际工程中，并可以根据实际工程应用要要增加相应功能。如机房温度监控、药品库房温度监控、粮库温湿度监控，在各检测点安置无线湿度传感器，并制定温

湿度阈值，接受温度并进行判断，根据采集温湿度的不同范围，并确定相应控制方式，比如湿度大于 50%，可以自动打开除湿机，温度高于 40 可以自动打开空调，以达到智能化管理。并可结合历史温湿度，做出应急措施，以保证粮食的最佳存储。此外，在实际应用中可以增加各类数据传感器，结合无线模块地址设置和频道选择，结合 TDMA，FDMA 以及 CDMA 等网络通信原理以实现无线传感器网络。

实验准备：

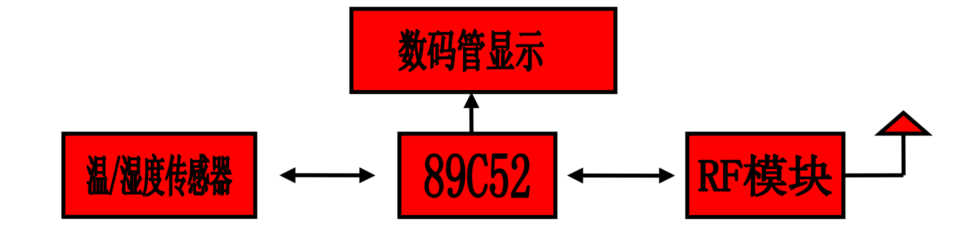
(1)硬件：JASk2132 开发板 1 套

(2 软件：Keil，ADS1.2，串口助手软件

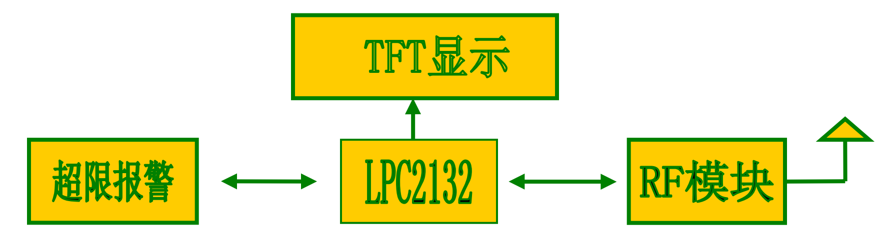
(4)开发工具：J-LINK,USB-TTL 模块，串口线

基本工作流程：

A:采集端



B:接收监控端



发送端，采用 DS18B20 温度传感器，并通过单片机 89C52 对 DS18B20 进行温度采集，采集完后单片机对采集的数据进行转换成有效数字信息，并通过无线模块将采集得到的温度数字数据发送出去；

接收端，通过无线模块接收发送端发送出的温度信息，当数据接收完成后产生接收完成中断信号，单片机确认有中断信息后读取无线接收缓冲区中的数据，根据数据包协议将接收到的信息通过数码管动态扫描方式显示当前温度，同时也将温度信息通过 232 串口上传给 PC 机，并对温度信息进行数据统计和数据存储。以下是上位机界面效果

操作步骤：

1.程序烧写

将该案例开发包配套的 51-JASK2000 代码下载到 JASK2000；

将该案例开发包配套的 ARM-JASK2132 代码下载 JASK2132 开发板中；

2.硬件连接

A:无线模块与 JASK2000 中 RF 接口对接后上电

B: 按照 PC--USB/TTL 模块--JASK2132 方式连接

3.检测

按照打开温度采集检测软件，按照截图进行操作，可以看到电脑之间无线通信

详细请参考 JASK2132 配套光盘实验程序和说明



无线温度采集主界面



无线温度采集曲线

案例四:无线 AD 传感器应用

功能概述

虽然现在很多传感器已经都数字化，但模拟类传感器依然大量存在，LPC2132 拥有 1 个 10 位 8 路 AD 转换器，启动 AD 转换方式很灵活，既可以单路采集，也可以审核制 BURST 方式对多路信号逐次循环采样，为构建无线传感器网络打下基础。

实验准备：

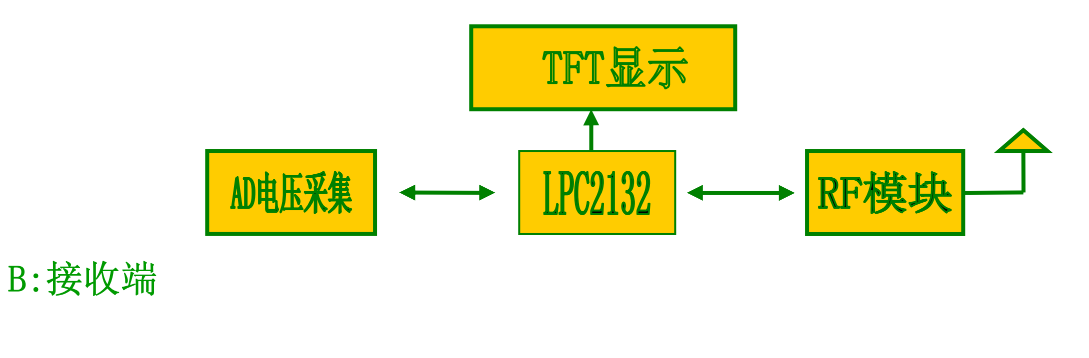
(1)硬件：JASk2132 开发板 1 套

(2 软件：Keil，ADS1.2，串口助手软件

(4)开发工具：J-LINK,USB-TTL 模块，串口线

基本工作流程：

A:采集端



B:接收端

数码管显示 ← 89C52 ↔ RF模块

发送端，LPC2132 通过片上 AD 功能采集电位器电压，采集完后将采集到数据转化为十进制数，并通过无线模块发送出去；

接收端，通过无线模块接收发送端发送出的温度信息，当数据接收完成后产生接收完成中断信号，单片机确认有中断信息后读取无线接收缓冲区中的数据，根据数据包协议将接收到的信息通过数码管动态扫描方式显示电压值大小。

操作步骤：

1.程序烧写

将该案例开发包配套的 51-JASK2000 代码下载到 JASK2000；

将该案例开发包配套的 ARM-JASK2132 代码下载 JASK2132 开发板中；

2.硬件连接

A：无线模块与 JASK2132 中 RF 接口对接后上电

B：无线模块与 JASK2000 中 RF 接口对接后上电

3.检测

按照打开温度采集检测软件，按照截图进行操作，可以看到电脑之间无线通信

详细请参考 JASK2132 配套光盘实验程序和说明

无线应用注意事项

(1) 无线模块的 VCC 电压范围为 1.8V-3.6V 之间，不能在这个区间之外，超过 3.6V 将会烧毁模块。推荐电压 3.3V 左右。

(2) 除电源 VCC 和接地端，其余脚都可以直接和普通的 51 单片机 IO 口直接相连，无需电平转换。当然对 3V 左右的单片机更加适用了。

(3) 硬件上面没有 SPI 的单片机也可以控制本模块，用普通单片机 IO 口模拟 SPI 不需要单片机真正的串口介入，只需要普通的单片机 IO 口就可以了，当然用串口也可以了。

模块按照接口提示和母板的逻辑地连接起来

(4) 标准 DIP 插针，如需要其他封装接口，或者其他形式的接口，可联系我们定做。

(5) 与 51 系列单片机 P0 口连接时候,需要加 10K 的上拉电阻,与其余口连接不需要。其他系列的单片机，比如 AVR,PIC，由于输入输出电流很大，如果是 5V 的，请参考该系列单片机 IO 口输出电流大小，如果超过 10mA，需要串联 2-5K 电阻分压，否则容易烧毁模块！如果是 3.3V 的，可以直接.

(6) 任何单片机都可实现对无线模块的数据收发控制，并可根据我们提供的程序，然后结合自己擅长的单片机型号进行移植;

(7) 频道的间隔的说明：实际要想 2 个模块同时发射不相互干扰，两者频道间隔应该至少相差 1MHZ，这在组网时必须注意,否则同频比干扰。

(8) 实际用户可能会应用其他自己熟悉的单片机做为主控芯片，所以，建议大家在移植时注意以下 4 点：

A:确保 IO 是输入输出方式，且必须设置成数字 IO;

B:注意与使用的 IO 相关的寄存器设置，尤其是带外部中断、带 AD 功能的 IO，相关寄存器一定要设置好;

C:调试时先写配置字，然后控制数据收发

D:注意工作模式切换时间

公司名称：杭州威步科技有限公司 联系电话：0571-86576692 18958007935
传真：0571-86576692 E-mail: wenming_hu2002@yahoo.com

最后，欢迎您使用我们的产品，产品在应用中有技术问题请及时向我们联系，我们会予以技术知道，同时运输中出现产品问题我们会全面责任并予以更换。

进步就是永不停步

愿与您一起走向成功！